



A3-D9

Personalized Portals – Personal Reader Framework, and Prototypes MyEar and MyNews

Project title:	Reasoning on the Web with Rules and Semantics
Project acronym:	REWERSE
Project number:	IST-2004-506779
Project instrument:	EU FP6 Network of Excellence (NoE)
Project thematic priority:	Priority 2: Information Society Technologies (IST)
Document type:	D (deliverable)
Nature of document:	P (prototype)
Dissemination level:	PU (public)
Document number:	IST506779/Hannover/A3-D9/D/PU/b1
Responsible editors:	Nicola Henze
Reviewers:	Arne Kösling
Contributing participants:	Hannover
Contributing workpackages:	A3
Contractual date of deliverable:	February 28, 2007
Actual submission date:	March 18, 2007

Abstract

This report documents the delivery of several prototypes which enable personalized access to information in the Web. We report on the progress of our framework for designing, implementing, and maintaining portal-like applications which syndicate and personalize the access to information in the Semantic Web, and two of the latest prototypes: *MyEar: Personal Music Syndicator* and the *MyNews Syndication Service*. Both prototypes are online available. Animated presentations of the framework and video documentations of the prototypes have been developed and are online available, too.

Keyword List

personalization, semantic web, web services, MyEar, MyNews, Personal Reader Framework

Project co-funded by the European Commission and the Swiss Federal Office for Education and Science within the Sixth Framework Programme.

© REWERSE 2007.

Personalized Portals – Personal Reader Framework, and Prototypes MyEar and MyNews

Fabian Abel¹, Ingo Brunkhorst¹, Nicola Henze¹, Daniel Krause¹

¹ Research Center L3S &
IVS- Semantic Web Group , University of Hannover,
Appelstr. 4, D-30167 Hannover, Germany
{abel,brunkhorst,henze,krause}@l3s.de

March 18, 2007

Abstract

This report documents the delivery of several prototypes which enable personalized access to information in the Web. We report on the progress of our framework for designing, implementing, and maintaining portal-like applications which syndicate and personalize the access to information in the Semantic Web, and two of the latest prototypes: *MyEar: Personal Music Syndicator* and the *MyNews Syndication Service*. Both prototypes are online available. Animated presentations of the framework and video documentations of the prototypes have been developed and are online available, too.

Keyword List

personalization, semantic web, web services, MyEar, MyNews, Personal Reader Framework

Contents

1 Executive Summary

The work in working group A3 is centered around three axes: In the first axis, we research foundations for personalization and adaptation in the Semantic Web, and in particular aim at logical frameworks for describing and characterizing appropriate personalization functionality. This axis is therefore called *Adaptive Functionality*. The second axis is on deploying personalization functionality in systems and prototypes – the *Testbeds*-axis. In the third axis, we develop a framework for designing and maintaining portal-like applications which syndicate and personalize the access to information in the Semantic Web.

This report belongs to the third axis and summarizes our results in developing an appropriate framework and prototypes.

The deliverable reports on

1. The framework which has been developed to design, develop and maintain Web syndication applications, the *Personal Reader Framework* (see Section ??)
2. Two of the latest prototypes:
 - (a) the *MyEar Personal Music Syndicator* (see Section ??), and
 - (b) the *New Syndication Service* (see Section ??).

2 The Framework: Personal Reader

Our approach for a Semantic Web browsing offers users a uniform entry point to access information, and in particular to Web services in the Semantic Web. It has been realized as part of the *Personal Reader Framework* which offers an environment for designing, implementing and realizing Web content readers in a service-oriented manner (see Figure ??):

- Web services deliver personalized recommendations for content, extracted and obtained from the Semantic Web. Using Semantic Web techniques they describe their offered functionality in a machine processable format. Optionally, they can return visualization templates that can be used to create user interface snippets for presenting the results of the Web services. We call these kind of Web services *Personalization Services*, *PServices* for short.
- *Meta PServices* can be employed to have single entry points to cooperating or concurrent PServices. For example, a music recommender Meta Personalization Service orchestrates different PServices delivering personalized music recommendations. In an internal processing step the music Meta-Web service filters and orders all the resulting recommendations of the different PServices into one single result.
- For syndicating the results of PServices and for creating appropriate user interfaces, *Syndication Services* (*SynServices* for short), are responsible for displaying the results of several PServices and/or Meta PServices to the user. Each SynService provides such an user end point to a certain domain or task, and allows the user to benefit from many PServices simultaneously, which are selected, combined and customized as the user wishes. SynServices can be realized as RDF browsers, can implement their own RDF processing interface,

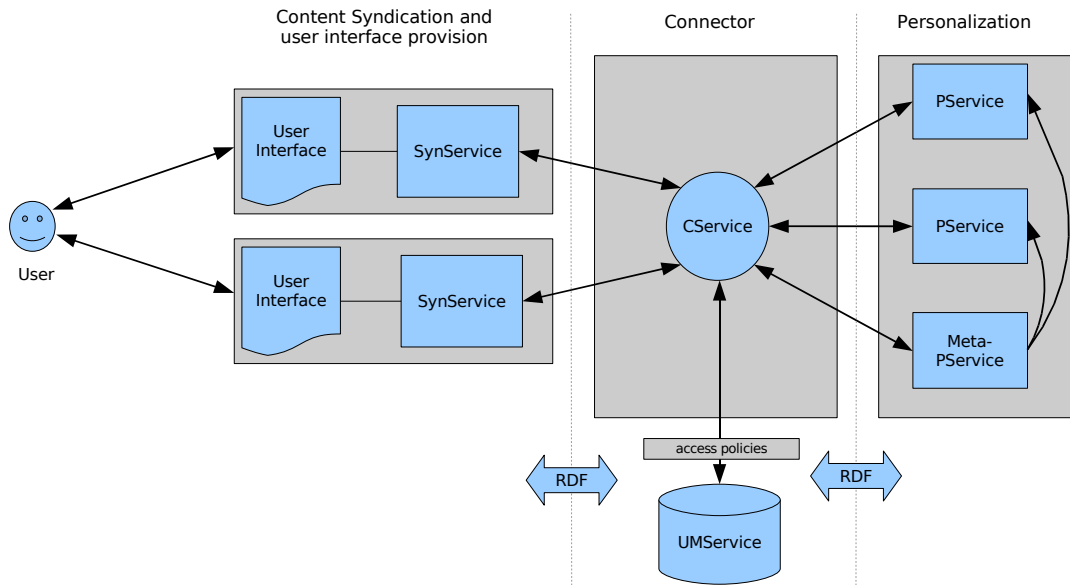


Figure 1: Overview of Personal Reader architecture

or they can make use of visualization templates provided by the different PServices / Meta PServices.

The *MyEar Music Syndicator* is an example of such a SynService, which provides the user end point to specify requests for podcasts. Throughout the paper, we use the MyEar Music Syndicator to illustrate our ideas of browsing information in the Semantic Web. Other examples of SynServices provide a user end point for viewing learning objects in an embedding context (where each aspect of this embedding context – like recommendations for quizzes, for examples, for further details with respect to the learning object’s content, etc., is provided by different PServices), or a user end point for browsing scientific publications of a large European project .

- For enabling the whole process, a core information provider – a user modeling service, *UMService* for short, deriving appropriate user profiles – is essential. In our architecture, the *UMService* realizes a centralized approach for user modeling, maintaining and protecting information about a user on behalf of this user. The main reason for choosing a centralized approach is to realize privacy protection: the user has full control about his user profile, and can define policies on which parts of this user profile might be public, or are available to trusted parties only. One central instance for all this information makes it possible to create awareness about stored information, and on realizing policy-protected access. Furthermore, this central *UMService* receives updates from SynServices or PServices in different domains, and allows for cross-domain re-use of user profile information (if the user wants that).
- From a technical point of view, another component is required to maintain the communication between the SynServices providing the user interface, the PServices, and the

UMService. This is the so-called *Connector Service* (*CService* for short) which harvest Web service brokers, collects information about detected PServices (for discovery, selection, customization, and invocation), and for organizing the communication between all involved parties, including requests to the UMService.

2.1 Using the Personal Reader Framework

An interaction with the system starts by the user who specifies her/his current request by formulating a query. For the matchmaking between user's query and provided functionality of detected PServices, the CService offers functionality for the required matching. The goal is to discover PServices that can incorporate and adapt best to the provided user data from both, the actual user's request and user profile data. N.B.: Not all PServices need to receive the same user profile data, as some of them might be trusted more than others. The necessary negotiation based on the user-defined policies in the UMService and credentials of the PServices have to be executed beforehand.

The discovery of PServices is done as following: First, the Syndicator searches for Web services having the required input and output parameters (*exact match*). This search is send as a request to the CService, which executes the search. If the exact match does not return any results, the CService can do a *plug-in match*, returning Web services that require more input parameters than given. These additional parameters have to be entered by the user in the second step, the configuration step. If the plug-in match cannot find appropriate Web services, *subsumed match*, *subsumed-by match* or a *best-match* can be performed. Which of the matching strategies shall be applied, and which order is preferred, is of course user dependent, the default process will take the order exact – plug-in – subsumed – subsumed-by – best.

Afterwards, all matching PServices will be displayed to the user who can choose which Web service(s) shall be invoked. With this selection step, it is ensured that only Web services are invoked that a user trusts, and negotiations about user profile credentials can be controlled by the user if necessary. Afterwards, PServices' customization parameters – if PServices offer them – are displayed to the user who can adjust them according to his requirements.

Every selected and customized PService is executed and returns its content, plus optionally one or several visualization templates. The visualization templates enable the SynServices to reach a high usability by providing domain-optimized visualization. Default visualization strategies are always available by using a RDF browser. The user can interact with the PServices by clicking on links or completing forms in the generated user interface. As these interactions are sent back to the PService it can adapt it's content more precise to the user's requirements and deliver more personalized content, for example displaying a higher level of detail of the relevant informations.

If a PService detects patterns of usage from the user interaction or certain user requirements, it can send an update request to the UMService. Again, not all PServices are allowed to update the user profile, and the update requests are distinguished according to the overall credentials of a PService, and the credentials of the reasoning process used by a PService to interpret user information. The UMService can allow or deny update requests on behalf of the user. Again, the centralized approach shows its benefits, and approved user profile updates are available immediately to not only the current SynService but to all SynServices for further, user-optimized presentation of Web content.

2.2 Visualization and Interface

All user interaction is realized via special Syndicator Services (SynServices). A SynService provides the interface for searching and configuring Web services, as described above. After Web services were selected, configured and invoked, the SynService displays the results of all PServices which have been invoked and returned results. This separation of content collection and syndication / visualization ensures an easy processing of the PServices' output, and it allow the SynServices to adjust visualization according to user devices' capabilities and limitations, or further user preferences.

By delivering visualization templates, every PService can optimize visualization and usability, as certain domain-specific information can be taken into account for creating the user interface.

2.3 Advantages of the Personal Reader Approach

A very important issue for improving the usability of Semantic Web browsers is adequate and intuitive visualization of content and it's access. A common approach in today's Semantic Web browsers is to visualize raw RDF files while disregarding the domain and purpose of the RDF document. Therefore, usability of such domain independent Semantic Web browsers is weak. Other approaches focus on some specific domain, and may obtain high usability for this single domain. This procedure has the disadvantage that domain spanning-browsing or domain-spanning content processing is not possible. Consequently, the user has to switch browsers whenever he switches between domains. Synergy effects based on a single user interface and domain spanning user profiling cannot be achieved.

Our approach combines domain specific visualization on the one hand, and a single user interface creation and user profile maintenance on the other. Therefore, we out-source visualization from the Semantic Web browser to the PServices by letting the PServices specify domain-specific and optimized visualization strategies, available via visualization templates. Visualization templates are used by SynServices for creating the final user interface, and – additionally – can optimize the user interface to match the constraints of the currently used device and user specific preferences. To ensure that also results of PServices, which do not provide some visualization template, can be displayed, each SynService implements a generic visualization of RDF documents based on a RDF browser like Piggy Bank¹. With this dual approach, a domain-optimized user interface will be provided whenever possible, and, as a fall-back solution, general-purpose RDF visualization is possible. This approach has the advantage that domain modeling has not to be done twice (in the Semantic Web browser and in the PServices); furthermore, changes in the output or the visualization of the PService do not require changes in the Semantic Web browser. The user directly interacts with the visualization from the PService, thus PService-specific interactions are enabled, too.

A further advantage of the architecture of the Personal Readers is that access to the user profile can be restricted to trustworthy instances only. This enables high usability as the same information do not need to be given manually to every single Web service, and privacy protection strategies can be facilitated under the full control of the user. By enabling user profile updates from PServices, domain-specific user modeling techniques can and should be implemented in the PServices. The very same argument as in the case of visualization also applies here: domain-specific information should be maintained by those parties which naturally have access to it,

¹<http://simile.mit.edu/piggy-bank/>

other approaches always require duplication or at least exchange of domain models which is error-prone, and at least time consuming.

2.4 Animated Presentation

An animated presentation of the personal reader architecture is available online at

<http://personal-reader.de/presentation/reverse07/>

3 MyEar

Scenario:

Assume a user who searches for podcasts in the Web. He enters a query and receives a list of appropriate podcast delivery services. He specifies which of these services he wants to launch. The user gets a list of all mandatory and optional parameters which can be used to tailor the services – the MyEar Syndication Service tries to fill all these parameters according to the information it has about the user’s preferences. The user can change or simply approve these parameters, eventually the user is requested to enter information that the MyEar Syndication Service was not able to provide. Finally, the user gets the syndicated output of all the services he launched, displayed in his personal Web interface. The appropriate visualization is chosen with respect to the currently used display device of the user.

The Personal Reader Agent acts as a portal to the Personal Reader infrastructure (see Figure ??). It allows the selection of appropriate Syndication Services.

In the first step the user can select a Syndication Service which is discovered by the Personal Reader Agent. In Figure ?? the user has the ability to select either the MyEar Syndication Service, which provides personalized Podcasting Feeds, or the MyNews Syndication Service, which provides personalized News Messages.

In the second step, the user can configure selected PServices. For example, the MyEar Syndication Service allows the user to specify keywords, duration and iTunes category of the podcasts he wants to listen to. The description of these customization parameters is provided by the PServices. The user profiling which enables the automatic configuration of the PServices is at the moment a simple one: It stores the parameters the user has entered the last time he used this Web service, and returns them as the default selection in the configuration dialog (Figure ??).

The user has the opportunity to store configurations of Personalization Services so that she/he or other users can re-use them (see Figure ??).

After configuration, the MyEar Syndication Service is invoked with the specified parameters. This invocation is passed - via the connector - to the corresponding PServices and MyEar receives the determined content (coded as RDF), as well as visualization templates. Only one visualization template is currently available, which displays the RDF document on PCs within a Web browser, as can be seen in Figure ?. The possibility to provide visualization templates by the PServices allows for domain-specific optimization of the user interface, which is not realizable with general-purpose RDF browsing approaches. In the case of the MyEar Syndication Service, for example drag and drop operations are available for selecting podcasts and controlling the audio together with further, music domain-specific gadgets.

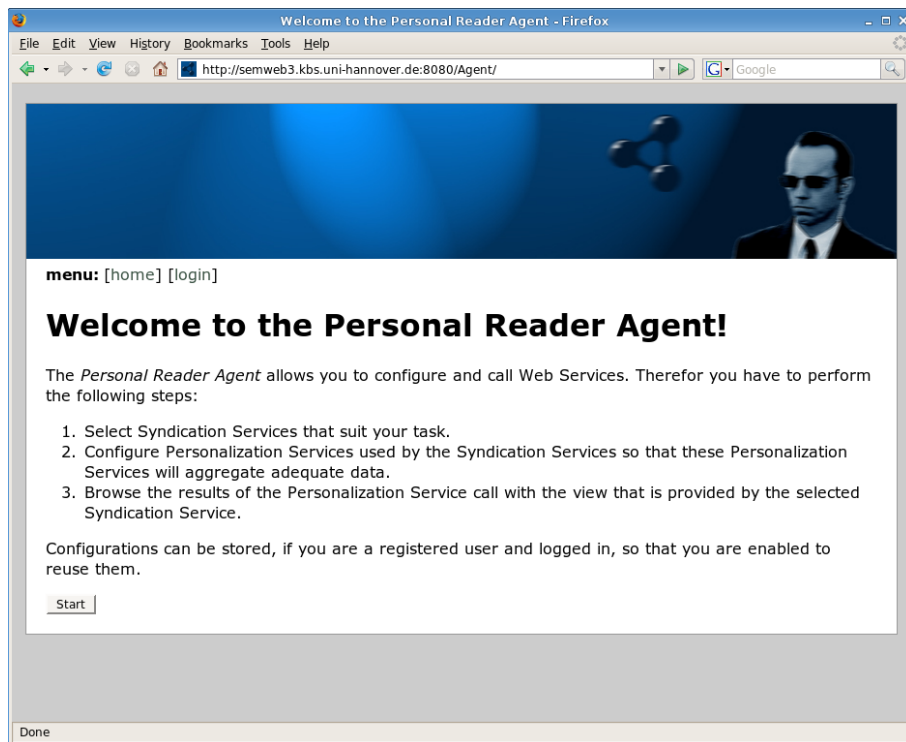


Figure 2: Personal Reader Agent = Portal for the Personal Reader Infrastructure

The MyEar Player further allows the user to manage playlists or listen to plalists of other users (friends) (see Figure ??).

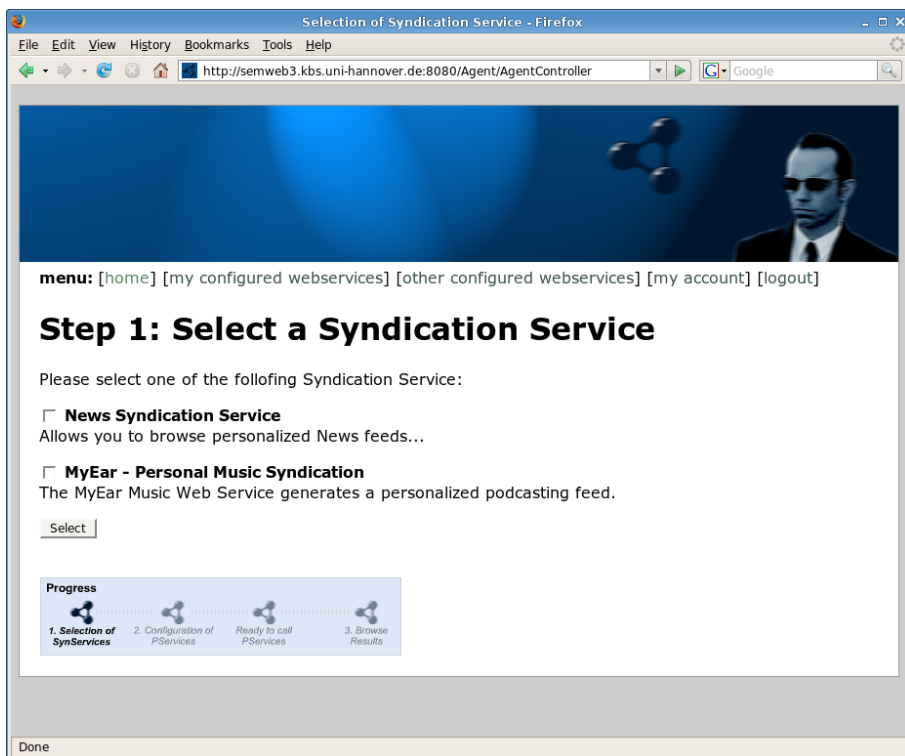



Figure 3: Selection of Syndication Services

Configuration of Syndication Service - Firefox

File Edit View History Bookmarks Tools Help

http://semweb3.kbs.uni-hannover.de:8080/Agent/AgentController

Google



menu: [home] [my configured webservises] [other configured webservises] [my account] [logout]

Step 2: Configuration of Personalization Service

Selected Syndication Service: MyEar - Personal Music Syndication
Contact: Personal Reader Team

Configure the Personalization Services, that are used by the selected Syndication Service, to your needs...

MyEar Configuration

Configurable Things of MyEar Service

itunes category

An itunes:category... [Info: This restriction may lead to few considered items or in extreme case also to timeouts!]

-- none --

Maximum Number of Google API Calls

To limit the waiting time you can select a maximum number of Google API Calls.

3
 2
 4
 1

Duration

Specify a range for the length of the audio files. If you are searching for radio-like shows then you should enter a high Minimum Duration. Otherwise if you are searching for single songs then we recommend you to enter a low Maximum Duration.
Info: Some podcasting producer do not specify the duration of their audio files. If you do not want to pass over these podcasts then leave both fields blank.

Minimum Duration
The minimum duration of audio files (in minutes).
-- none --

Maximum Duration
The maximum duration of audio files (in minutes).
-- none --

Query Keywords

Type in your keywords (seperated with blanks) that should be within the podcasting item, e.g.: Jazz Swing

Value:

Enter at least 1 values.

Done

Go back to selection of Webservices: [back](#)

Progress

1. Selection of SynServices **2. Configuration of PServices** 3. Ready to call PServices 3. Browse Results


Done

8
Figure 4: Configuration of Personalization Services

Configuration of Syndication Service - Firefox

File Edit View History Bookmarks Tools Help

http://semweb3.kbs.uni-hannover.de:8080/Agent/AgentController



menu: [home] [my configured webservices] [other configured webservices] [my account] [logout]

Step 2: Validate your Configuration

Selected Syndication Service: MyEar - Personal Music Syndication
Contact: Personal Reader Team

As a logged in user you have three opportunities:

- Click on the "back"-Button to edit the configured values
- Click on the "confirm"-Button to confirm your entries
- Click on the "confirm + save"-Button to confirm and save your entries. Saving allows you to re-use your configuration at a later date.

MyEar Configuration - Values you entered/selected

itunes category	Music
Maximum Number of Google API Calls	2
Duration	Minimum Duration: 15 Maximum Duration: 60
Query Keywords	Jazz

If you click the *confirm*-Button the Webservices you selected and configured are executed. Otherwise click the *back*-Button to edit your inputs.

Save your Configuration

Saved configurations can be used at a later date to call the Webservice in a personalized way. It is also possible to adjust your configuration later.

You have to enter at least a significant name for your configuration. Further you can enter a short description and select whether other users can utilize your configuration or not (standard is not).

Name: (e.g. MyEar - Jazz)

Description: (e.g. With this configuration the MyEar-Webservice returns a Jazz-Podcasting Feed...)

publish?: (if checked then other users can utilize your configuration)

Progress

1. Selection of SynServices 2. Configuration of PServices **Ready to call PServices** 3. Browse Results

Done

Figure 5: Selection of Syndication Services

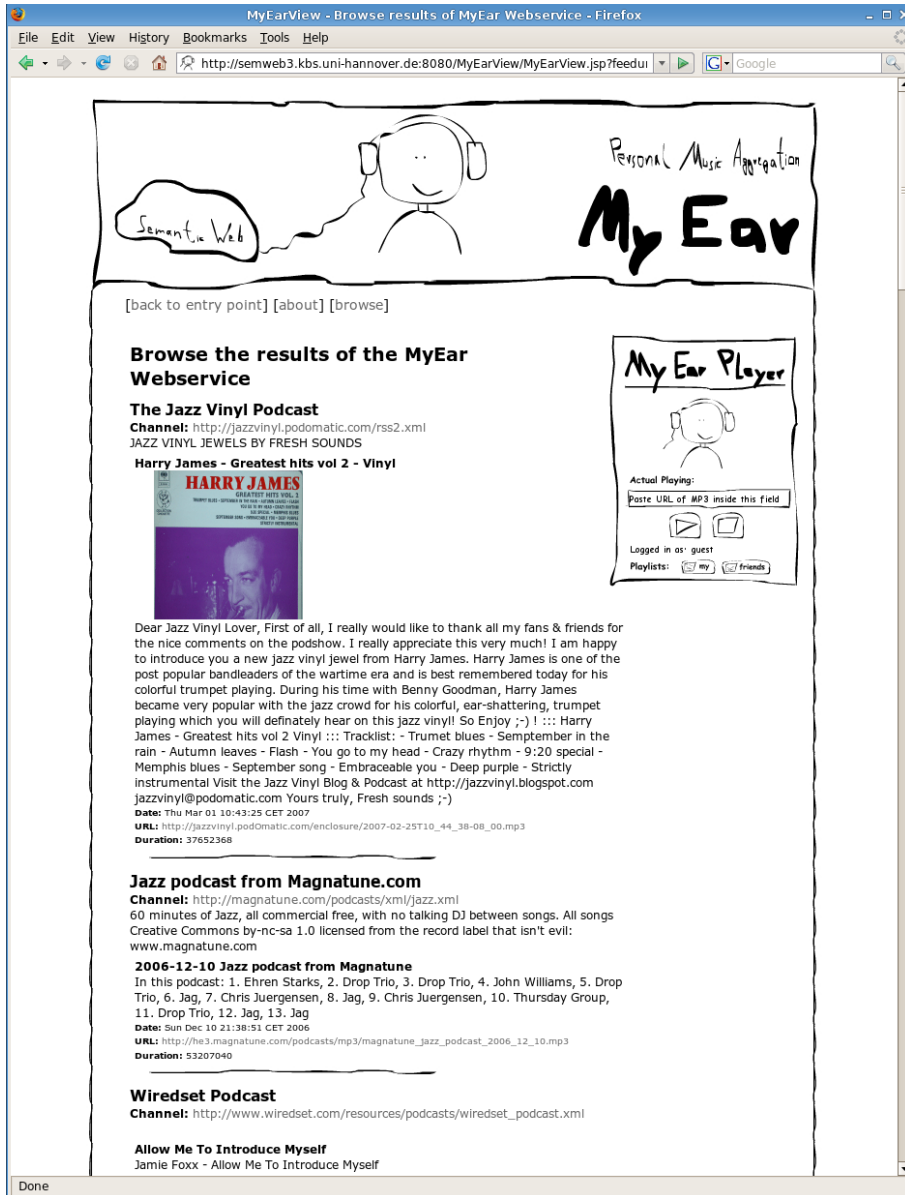


Figure 6: Results of the MyEar Personalization Service



Figure 7: Managing playlists and re-using playlists of friends

3.1 Prototype

Web page with additional information and access to the prototype and video documentation:

<http://www.personal-reader.de/agent/>

The prototype can be directly accessed at

<http://semweb3.kbs.uni-hannover.de:8080/Agent/>

with username = guest, password = guest.

Select the *MyEar - Personal Music Syndication* in the first step

3.2 Video Documentation

Available at <http://www.personal-reader.de/agent/videos/agent-full-video.html>

4 MyNews

The MyNews Service is a running demo application based on the Personal Reader Framework. It contains the News Personalization Service, the MyNews Syndication Service, and a visualization. The News Personalization Service accesses multiple RSS Feed sources and aggregates the feeds. In a second step it personalizes the results according to a users specifications by selecting appropriate news according to a user's interests. A screenshot is depicted in Figure ??.

The MyNews Syndication Service configures the invocation of the News Personalization Service by interacting with the user. Furthermore, it enriches the result of the News Personalization Service by parsing its result and invoking other Personalization Services, for example an image Personalization Service. This image Personalization Service delivers images that fit to the keywords extracted from the News Personalization Service response.



Figure 8: The MyNews Application

4.1 Prototype

The prototype can be directly accessed at

`http://semweb3.kbs.uni-hannover.de:8080/Agent/`

with username = guest, password = guest.

Select the *News Syndication Service* in the first step.

5 Conclusion

This deliverables documents on the delivery of prototypes for personalized access to Semantic Web data: the *MyEar - Personal Music Syndicator* and the *News Syndication Service*. Both prototypes have been designed and implemented by aid of the personal reader development framework, which itself has been further refined and extended during the reporting period. In this report, we give a summary of the Personal Reader Framework and describe the developed prototypes. The prototypes themselves are online available, together with accompanying descriptions, video materials, and presentations.

6 Acknowledgment

This research has been co-funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Program project REVERSE number 506779 (cf. <http://reverse.net>).

7 Appendix: Published Articles

The appendix contains two technical reports and ten published articles on the Personal Reader Framework and its applications during the reporting period in 2006.

1. Nicola Henze and Daniel Krause: Scalable Matchmaking for a Semantic Web Service based Architecture - Workshop on Semantics for Web Services, December 4, 2006, Zurich, Switzerland, collocated with ECOWS 2006.
 2. F. Abel, I. Brunkhorst, N. Henze, D. Krause, K. Mushtaq, P. Nasirifard and K. Tomaschewski: Personal Reader Agent: Personalized Access to Configurable Web Services. ABIS 2006 - 14th Workshop on Adaptivity and User Modeling in Interactive Systems, Hildesheim, October 9-11 2006.
 3. Nicola Henze and Daniel Krause: User Profiling and Privacy Protection for a Web Service Oriented Semantic Web. ABIS 2006 - 14th Workshop on Adaptivity and User Modeling in Interactive Systems, Hildesheim, October 9-11 2006.
 4. Nicola Henze and Daniel Krause: Personalized Access to Web Services in the Semantic Web. SWUI 2006 - 3rd International Semantic Web User Interaction Workshop, November 6, 2006, Athens, Georgia, USA, collocated with ISWC 2006
 5. Nicola Henze: Personalisierbare Informationssysteme im Semantic Web. In T. Pellegrini, A. Blumauer (Hrsg.): Semantic Web: Wege zur vernetzten Wissensgesellschaft. Springer, 2006, ISBN 3-540-29324-8.
 6. Lilia Cheniti-Belcadhi, Nicola Henze, Rafik Braham: Implementation of a Personalized Assessment Web Service. 6th IEEE International Conference on Advanced Learning Technologies, ICALT 2006, July 5-7 2006, the Netherlands.
 7. Nicola Henze: Personalized e-Learning in the Semantic Web. International Journal of Emerging Technologies in Learning (iJET), Vol. 1, No. 1 (2006).
1. Technical Report: Personal Reader Agent
 2. Technical Report: UMService: An application independent User Modelling Service

Scalable Matchmaking for a Semantic Web Service based Architecture

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

Abstract

We propose a two-step matchmaking procedure for a Web Service oriented architecture to cope with scalability problems if a large amount of Web Services has to be processed. We distinguish between domain-aware and domain-independent matchmaking, and show how such a two-step matchmaking process can be realized. We discuss the approach within the Personal Reader architecture which enables the use of Web Service based applications to personalize Semantic Web content.

1. Introduction

While more and more data became machine readable over the last years, the Semantic Web Stack [1] does not include any kind of application layer that uses these data to base applications on them.

At this point of time the usage of Web Services is the most promising approach to fill up this gap in the Semantic Web Stack. The main advantages of Web Services are their platform and location independence. Web Services are accessed via standardized Hypertext Transfer Protocol and therefore can be stored and used on every web server in the world. Web Services can build up their functionality utilizing Semantic Web content or other Web Services. By combining and syndicating different Web Services which offer basic functionalities, the realization of more complicated processes is possible. As every Web Service can physically be located on any web server in the world, the definition of the interfaces that need to be implemented require most attention.

Furthermore, as Web Services encapsulate functionalities, the reuse of functionalities – encapsulated in the Web Services – becomes feasible. The reuse of functionality by accessing Web Services is based on the assumption that Web Services are long-term available. But experiences from the WWW show that a network where many different people can interact in by creating own Web Services, is a quick changing environment. In the Semantic Web not only con-

tent changes quickly, but also Web Services that process the content might appear and disappear in a frequent manner.

Combining Web Services in such a dynamic environment requires that the combination process itself has to be dynamic. This means that Web Services have to discover other Web Services automatically, and need to be able to detect those Web Services that offer the momentarily required functionality. To support such a dynamic combination, the functionality of the Web Service has to be described *semantically* in a way that enables an automatic matchmaking between the requested and the offered functionalities.

In this paper we present the Personal Reader architecture that offers different kinds of Web Services: Syndication Services provide application features (like user interfaces and data syndication), Personalization Services provide personalization functionality (like e.g. the provision of contextual information, recommendations, etc.) in the Semantic Web. For describing the semantics of invocation and response parameters we introduce *configurable descriptions* which enable a two-step matchmaking to discover appropriate Web Services with a high efficiency and scalability.

2. The Personal Reader Architecture

Our approach for a Semantic Web architecture offers users a uniform entry point to access the Semantic Web, and in particular to Web services in the Semantic Web. It has been realized as part of the *Personal Reader Framework* [2, 3] which offers an environment for designing, implementing and realizing Web content readers in a service-oriented manner (see Figure 1):

- Web services deliver personalized recommendations for content, extracted and obtained from the Semantic Web. Using Semantic Web techniques they describe their offered functionality in a machine processable format. Optionally, they can return visualization templates that can be used to create user interface snippets for presenting the results of the Web services. We call these kind of Web services *Personalization Services*,

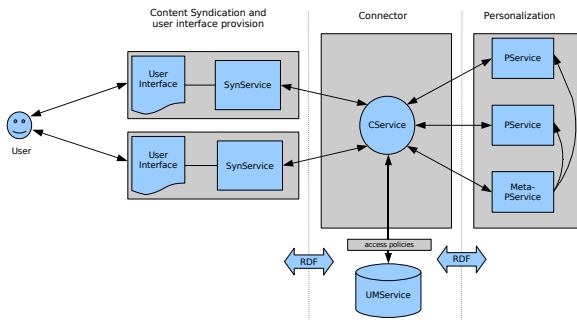


Figure 1. Overview of Personal Reader architecture

PServices for short.

- *Meta PServices* can be employed to have single entry points to cooperating or concurrent *PServices*. *Meta PServices* offer the technical platform in the Personal Reader architecture to orchestrate simple and generic *PServices* to model even complicated domains or relationships.
- For syndicating the results of *PServices* and for creating appropriate user interfaces, *Syndication Services* (*SynServices* for short), are responsible for displaying the results of several *PServices* and/or *Meta PServices* to the user. Each *SynService* provides at least one user end point (that is also called user interface) to a certain domain or task, and allows the user to benefit from many *PServices* simultaneously, which are selected, combined and customized as the user wishes. *SynServices* can be realized as RDF browsers, can implement their own RDF processing interface, or they can make use of visualization templates provided by the different *PServices* / *Meta PServices*.
- User interfaces are responsible for the interaction with the user. They receive XML or RDF messages from the *SynService* and visualize them according to the display device. Every user interface deals with one special class of devices, for example PCs, PDAs or mobile phones which means that device adaption is dealt with by the user interfaces. Therefore, one *SynService* can have many different user interfaces. The user can interact with the user interface by generating events, like clicking on a button, entering text in a form etc. These events are sent back to the *SynService* that processes these events.
- For enabling the whole process, a core information provider – a user modeling service, *UMService* for short, deriving appropriate user profiles – is essential.

In our architecture, the *UMService* realizes a centralized approach for user modeling, maintaining and protecting information about a user on behalf of this user. The main reason for choosing a centralized approach is to realize privacy protection: the user has full control about his user profile, and can define policies on which parts of this user profile might be public, or are available to trusted parties only. One central instance for all this information makes it possible to create awareness about stored information, and on realizing policy-protected access. Furthermore, this central *UMService* receives updates from *SynServices* or *PServices* in different domains, and allows for cross-domain re-use of user profile information (if the user wants that).

- From a technical point of view, another component is required to maintain the communication between the *SynServices* providing the user interface, the *PServices*, and the *UMService*. This is the so-called *Connector Service* (*CService* for short) which harvest Web service brokers, collects information about detected *PServices* (for discovery, selection, customization, and invocation), and for organizing the communication between all involved parties, including requests to the *UMService*.

2.1. Usage Scenario

To describe the relationship between the different kind of web services we point out which steps need to be performed until the user gets his personalized content visualized in the user interface:

First, the user logs in at the *UMService*, using his username and password. Then *UMService* creates a session ID (SID for short) which will be valid during the whole session, and can only be traced back to the user at the *UMService* (realized via public-private-key encryptions). After logging in, the user accesses an entry point. The *SynService*, that belongs to the entry point, calls the *CService* and receives a list of available *SynServices*, a human readable description and the URL of the user interface that is appropriate to the user's display device. Entry points can give different levels of support to make the selection of the *SynServices* more easy for the user. Possible entry points are:

- The entry point can be an agent and ask the user what task he wants to fulfil and to recommend *SynServices* that cope with this issues.
- Another entry point can build a hierarchical organized portal, grouping similar *SynServices*.
- A third entry point may just returns an unordered list of all available *SynService*.

The user selects the SynService that fits his tasks best and is redirected to the URL of the user interface that belongs to the selected SynService which is able to visualize the content according to the user's display device. Furthermore, by accessing the user interface, the SID is passed to it and along to the SynService. The SynService requests a list of available PServices from the CService. In this request the SynService also submits which Ontologies it is able to handle. The CService does the first step of the matchmaking and returns PService candidates. In the second step of the matchmaking, the SynService selects PServices according to their description of offered functionality.

To invoke the selected PServices the SynService requires invocation parameters to personalize the results of the different PServices according to the user's preferences. These invocation parameters can be gained in two different ways:

- The SynService asks the UMService if it has stored information how the user has specified a value for a parameter beforehand. It is used in the confidence that this previous value will be valid for the current invocation. If the confidence is too low or there are no values stored until now, the user is asked directly.
- The SynService asks the user directly via the user interface to define a value for the required invocation parameter.

If both ways fail, for example if no information is stored in the UMService and the user rejects to enter a value, and the invocation parameter is marked as "required" in the description of the PService, this PService is omitted by the SynService. Afterwards, the SynService invokes the remaining PServices by sending an invocation request to the CService. The CService invokes every PService synchronous. The PServices use the invocation parameters and additionally, can send requests to the UMService to get more user specific data. Afterwards, they return the results to the CService that combines all single responses to one response and sends it back to the SynService. The SynService combines, filters and enriches the response and formats it in a way that user interfaces can easily visualize the response. This response is sent from the SynService to the user interface and is displayed to the user.

In an iterative manner the user or the SynService now can alter and refine invocation parameters to receive different or better results.

2.2. User Modeling

As illustrated in the previous chapter two types of Web Services interact with the user: The SynService as it gets input from the user via events and the PService that gets

personalized invocation parameter. This enables both types of Web Services to derive properties about the user with the help of different user modeling techniques. As both types of Web Services operate in one special domain they both should have detailed knowledge of the domain that they can take into account when modeling the user.

2.3. Semantic Web Services

To enable automatic matchmaking we require a machine-readable description of the functionality of our Web Services. Our approach relays on a semantic abstraction of a datatype-based description: instead of using datatypes like strings, integer, etc. to describe single parameters we group these parameters into semantic parameters that are described by Ontologies in a Configurable Description: For example, some parameters `keyword1, keyword2, ..., keywordn` build the semantic parameter `query` that is stored in the Configurable Description. By defining the ontology, that is used for the Configurable Description, precise enough, invocation and response parameters are sufficient to describe the functionality of the whole Web Service. Furthermore, as the Configurable Description contains the vocabulary that the Web Service is able to process, it is the groundwork for our two-step matchmaking.

3. Matchmaking

In a Web Service oriented architecture matchmaking between descriptions of Web Services and requirements is always domain aware. That means, that programs that do the matchmaking require domain knowledge. Thus, there are two possible realizations for matchmaking:

- *Centralized matchmaking*: One programs knows all domains that are used in the architecture.
- *Decentralized matchmaking*: For every single domain, and for all domains which are commonly used together, there has to be an own matchmaking program.

The first solution would run into problems if new Ontologies appear, because every new Ontology determines an update of the matchmaking system. Furthermore, this approach does not scale as more and more Ontologies appear, the program will result in a more and more complicated system which makes it unmaintainable.

The second solution scales very well in terms of the appearance of new Ontologies as existing systems have not to be changed but only new ones are added. The problem to cope with in this solution is the number of Web Services: The larger the number of available Web Services is, the longer the matchmaking process takes.

Our solution for this problem is to introduce a two-step

	Centralized matchmaking	Decentralized matchmaking	Two-step matchmaking
adding new domains	difficult	good	good
handling large amounts of Web Services	difficult	difficult	good

Table 1. Comparison of different match-making approaches

matchmaking: In the first step our central Web Service, the CService, does a domain-independent matchmaking by checking whether two Web Services can handle the same Ontology. This is done by analyzing the Configurable Description of the Semantic Web Services. All Web Services that are able to handle the required Ontology are marked as candidates and passed to the second step of the matchmaking process. This first step is very efficient as it is not more than a simple RDF database lookup in our Configurables database.

In the second step of matchmaking, we use a domain-aware matchmaking program like in the second realization. This is done by the SynServices, that know best which functionalities, expressed by the description of input and output parameters in the Configurable Description, are required.

This solutions scales in two ways: On the one hand the appearance of new Ontologies does not require changes on the centralized component. On the other hand, the first step, that scales very well in term of handling large amounts of Web Services, slashes the amount of Web Service that are passed to the second step. Therefore, the time that is required for the matchmaking process is little influenced by adding new domains to the system.

4. Discussion of the approach and related Work

The approach presented in this paper differs from common usage of Web Services, and shows how Web Services can be used as first class citizens in a Semantic Web. In our thinking, Web Services provide functionality snippets, which shall be plugged together to support a user during the task he is currently performing. Thus, Web Services provide functionality in the Web for *end users*, and our approach shows how user interfaces for accessing, plugging together, syndicating and using Web Services can be realized. The different stakeholder of the process are the Web Services which take care on appropriate user interfaces (Syndication Services), and those who provide the functionality snippets

(the Personalization Services). For performance and re-usability issues, a communication facilitator has been employed (the Connector Service).

Related work to our approach can be found in projects which apply Web services in the Semantic Web: Current research here focuses more on enabling technologies like Web services discovery, composition and orchestration (cf. [4, 5]). However, approaches which focus on usability and user-interfaces for accessing a Web Service-oriented Semantic Web are today missing.

5. Conclusion and Further Work

In this paper, we have proposed an approach to realize personalized access to Web Services in the Semantic Web. We have identified the main challenges to overcome, and especially discussed the influence of domain dependent vs. domain independent discovery, selection and invocation of Web Services. We are currently extending the architecture of the Personal Reader Framework as discussed in the paper to realize the full functionality of the configurable descriptions and the matchmaking processes. A prototype showing the applicability of the proposed solutions has already been realized and demonstrates the syndication of various podcast providers according to the music interests of users¹.

References

- [1] BERNERS-LEE, T. Semantic Web - Keynote at XML 2000 Conference, 2000. <http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html>.
- [2] HENZE, N., AND KRAUSE, D. Personalized access to web services in the semantic web. In *SWUI 2006 - 3rd International Semantic Web User Interaction Workshop, colocated with the 5th International Semantic Web Conference* (nov 2006).
- [3] HENZE, N., AND KRIESELL, M. 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).
- [4] MCILRAITH, S., SON, T., AND ZENG, H. Semantic web services. *Intelligent Systems* 16, 2 (2001), 46–52.
- [5] MOTTA, E., DOMINGUE, J., AND CABRAL, L. Irs-ii: A framework and infrastructure for semantic web services. In *Proceedings of the 2 Intl. Semantic Web Conference* (Florida, USA, 2003).

¹The prototype can be accessed via www.personal-reader.de/agent/

Personal Reader Agent

F. Abel, I. Brunkhorst, N. Henze, D. Krause, K. Mushtaq, P. Nasirifard and K. Tomaschewski
Information Systems Institute, University of Hannover
Appelstrae 4, 30167 Hannover, Germany
readerteam@kbs.uni-hannover.de

Abstract

The Personal Reader Framework enables us to develop and maintain Web Content Reader. In this architecture Web Services are primarily used as providers of RDF data, the content, which is presented the end user of a Web Content Reader. With our new approach of Configurable Web Services we allow users to configure the data providing Web Services. Such configurations can be stored and reused at a later date. Thereby the Personal Reader Agent is the interface between Users and Configurable Web Services. The Agent allows selection, configuration and calling of the Web Services and further provides personalization functionalities like reuse of stored configurations which suit to the users interests.

1 Introduction

Within the Personal Reader project we already developed Web Content Reader like the *Personal Publication Reader*[PPR, 2005] which allows browsing publications in an embedded context. We also utilized and extended the SWAD-E Semantic Portal software[Reynolds *et al.*, 2005] to provide a Personal Semantic Portal[SemPortal, 2005]. Whereas these approaches are fixed in terms of the type of data that is provided, we now introduce a more generic approach: *Configurable Web Services* and the *Personal Reader Agent*. The Personal Reader Agent is a Web Application which enables users to select, configure and call Configurable Web Services. These Semantic Web Services need a detailed description of how they can be configured and how they are accessible. According to this description the Personal Reader Agent generates an interface that allows to adjust the Web Services. Personalization functionalities, like reuse of stored configurations of Web Services which suit to the users interest, lead to an adaptive, personal Agent.

2 Personal Reader Agent

The Personal Reader Agent is on the one hand a kind of wizard that allows to select, configure and call Configurable Web Services and on the other hand it enables users to manage and reuse their saved configurations (*personalization functionality*).

2.1 Configurable Web Services

Each Configurable Web Service has a detailed RDF description which defines parameters that can be used to ad-

just the Web Service (*Configurable description*). An example is the *My Ear Music Web Service*: This service allows users to configure parameters like *music category* or *maximum duration of songs*. It results in a *podcasting feed* containing items that are aggregated from arbitrary feeds but fulfill the adjusted parameters. A formal definition of the Web Service's configurable parameters, thus a *Configurable Web Service*, has the advantage that the process of configuring the Web Service can be abstracted and further made configurations can be stored and reused. These to aspects are covered by the Personal Reader Agent.

2.2 Demonstration

Within a normal workflow the Personal Reader Agent affords the following steps:

- 1. Discovery and Selection** In this step the Agent requests human readable descriptions of the Configurable Web Services that are registered at our UDDI. Afterwards these descriptions are prepared for a selection by the user.
- 2. Configuration** After the first step the Agent reads in the Configurable descriptions of the selected Web Services and generates HTML Forms so that the user can perform the configuration (see figure 1).
- 3. Web Service Call** After all selected Web Services are configured without violating the restrictions defined in the corresponding Configurable descriptions (e.g. *maxNumberOfInputs*, *type*, ...), the Agent is ready to call the Web Services. In this step the user further has the opportunity to save the configuration (see figure 2).
- 4. Presenting the results** This step is not part of the Agent application but rather a task that can be done by a common RDF browser or an application that provides a special view for certain RDF data, e.g. *MyEar View* visualizes podcasting feeds (see figure 3).

Stored configurations and a corresponding user model build the fundament to enable users to...

...reuse their own configurations: In order to allow users a faster access to the Configurable Web Services they can call these services also with a saved configuration as illustrated in figure 4.

...reuse recommended configurations of other users: The Agent allows the listing of configurations that might be relevant for a user. To determine relevant configurations the Agent utilizes relations between users that are defined inside an Ontology that models persons and their involvements in working groups.

If two persons (*User A* and *User B*) are involved in the same working group then the Agent suggests that configurations made by *User A* are also interesting for *User B*.

3 Conclusion

With the Personal Reader Agent and Configurable Web Services we provide a dynamic approach to aggregate RDF data. The Agent's personalization functionality further allows to personalize the access to the Configurable Web Services.

At present the Personal Reader Agent is deployed as a prototype accessible via:
<http://www.personal-reader.de/agent/>

References

- [PPR, 2005] R. Baumgartner, N. Henze and M. Herzog The Personal Publication Reader: Illustrating Web Data Extraction, Personalization and Reasoning for the Semantic Web. *European Semantic Web Conference ESWC 2005*, Heraklion, Greece, 2005.
- [SemPortal, 2005] N. Henze and F. Abel. User Awareness and Personalization in Semantic Portals. *4th International Semantic Web Conference*, Galway, Ireland, 2005.
- [Reynolds et al., 2005] D. Reynolds, P. Shabajee, S. Cayzer and D. Steer. *Semantic Portals Demonstrator - Lessons Learnt*. SWAD-Europe deliverable 12.1.7, 2005.

A Figures

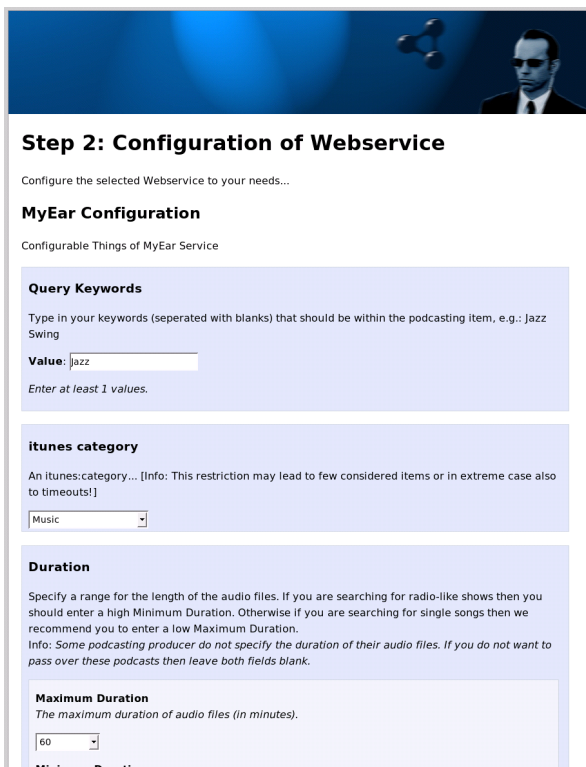


Figure 1: Step 2 - Configuration of Web Services

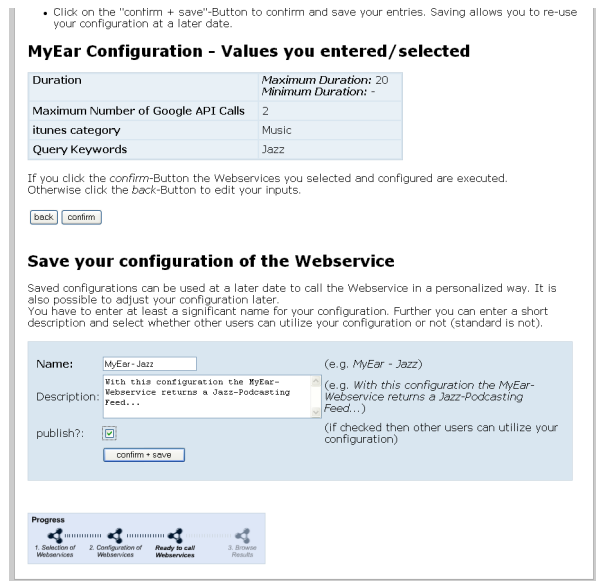


Figure 2: Saving Configurations - Entry of meta description about a configured Web Service

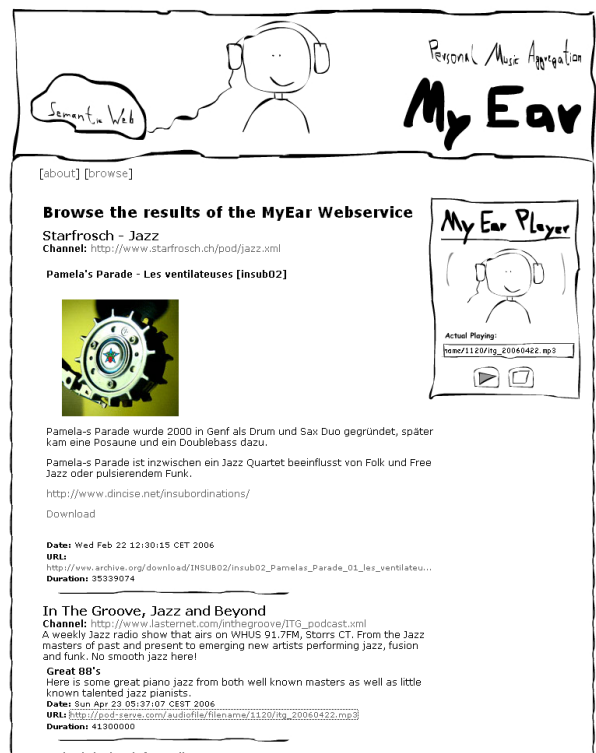


Figure 3: MyEar View

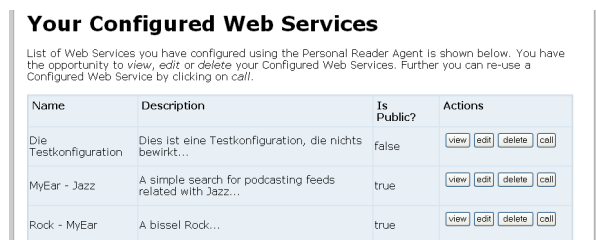


Figure 4: List of configured Web Services

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 normal case. 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 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 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 profiling
- Connector
- Personalization Services: Web Services that offer a certain personalization functionality

The syndication 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. The Connector handles the communication flow between all stakeholder in the architecture. The User Profile Manager 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 Personalization Services. This is realized by the central Connector, 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 Personalization Services are used by the Syndicator to generate a portal where users can select those Services which suit best their requirements. Thus, this portal represents a single access point to the available Web Services.

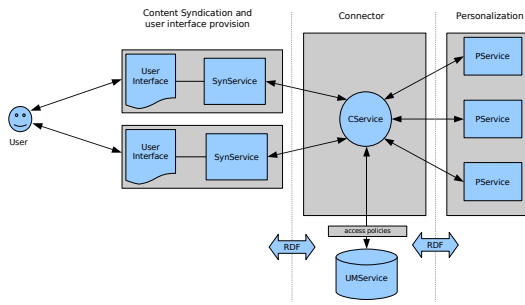


Figure 1: Simplified architecture of the Personal Reader Framework

Negotiation

Before a user can invoke the selected Web Services he selected from the portal, their invocation parameters must be set. The Syndicator tries to set these invocation parameters 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 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). 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)) ∧
        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 invocation parameter P' .

User interaction

If the access is denied, a Syndicator 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 Personalization Services are available whose invocation parameters can be automatically filled, use only those Services.
- 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 in order to adjust policies or not (with the fact of losing 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. 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 take care on the missing invocation parameters as described above. If all invocation parameters are configured, Web Services are invoked and their delivered contents are syndicated and visualized in an appropriate representation for the end user.

2.3 Collaborative Access to Adaptation Functionality

As the access policies in combination with the above defined ECA rule require user interaction, it can result in usability disadvantages if users often access new Web Services which they have not used before and automatic supply for user profile information fails. Our solution is that users can define other users they trust in. If these users U' allow a Web Service W to access their user profile to receive an invocation parameters P , the User Profile Manager 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)) ∨
        userProfile(U')] ∧
        readAccessAllowed(W,privacyProtection(P)) ∧
        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)) ∧
        [confidence(P,V) > threshold ∨
        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 apply user profile matching techniques to find similar users for collaboration.

2.4 User Profile Maintenance

User profile maintenance is handled by the User Profile Manager. Furthermore, the User Profile Manager restricts access to the sensible user profile information for unauthorized Web Services. 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 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, the access to these data is denied.

Our approach allows to store user profiles in a centralized place. So, every Web Service can access the user profile via the User Profile Manager. 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 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 its own user modeling techniques to derive new information about the user, and write the results directly to the central user profile. 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. 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?

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, 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. 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 Bank¹, Longwell² or Brownsauce³. 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 fully functional user profile. Our current work focuses on extending the so far implemented user profile to support all above described features. Future work will be the integration of more Personalization Web Services into our Personal Reader Framework to demonstrate the advantages of a shared user profile in the Semantic Web.

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 Fourteenth GI- Workshop on Adaptation and User*

Modeling in Interactive Systems (ABIS 06), Hildesheim, Germany, 2006.

- [Bailey *et al.*, 2004] James Bailey, George Papamarkos, Alexandra Poulouvassilis, and Peter T. Wood. An event-condition-action language for xml. In Mark Levene and Alexandra Poulouvassilis, 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 European 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.

¹<http://simile.mit.edu/piggy-bank/>

²<http://simile.mit.edu/longwell/>

³<http://brownsauce.sourceforge.net/>

Personalized Access to Web Services in the Semantic Web

Nicola Henze and Daniel Krause

IVS - Semantic Web Group, University of Hannover,
Appelstr. 4, D-30167 Hannover, Germany
{henze,krause}@kbs.uni-hannover.de

Abstract. We propose a Semantic Web browser which enables users to discover, select and personalize Web services in the Semantic Web. We discuss how such a Semantic Web browser can offer domain specific visualization, and investigate solutions for supporting individual users in finding appropriate Web services, in tailoring Web services to their needs, and in accessing and administering their favorite Web services. As a proof-of-concept, we show the realization of a Semantic Web browser, the personal podcast syndication service *MyEar*, which collects and filters data from various providers of podcast feeds.

keywords: semantic web, semantic web browser, personalization, web services

1 Motivation

The Semantic Web aims at enabling a huge, dynamic and federated network of media entities and information, enriched with machine-processable semantics. Many of today's research initiatives in the Semantic Web focus on developing technologies, languages, reasoning languages and tools for realizing an appropriate backbone for the Semantic Web vision. While these enabling technologies are being developed, the question of how to assist users to benefit from the emerging Semantic Web, thus on how to realize the second part of the vision by T. Berners-Lee et al. [2]: "better enabling computers and people to work in cooperation" currently receives less attention.

To realize one of the most promising advantages of the Semantic Web – the possibility to personalize output according to user's needs – Semantic Web services have to cope with three user-centered issues:

- allowing users to specify their needs (customization)
- optimizing result evaluation according to explicit and implicit needs of the user (adaptation; explicit needs are directly obtained during a particular interaction, implicit needs are derived from previous interactions and are interpreted and consolidated by aid of a user modeling component)
- presenting their results in a way that
 - user-side applications can visualize the results

- transparency and controllability of the result-determining processes and the adaptation steps are guaranteed.

The paper is organized as follows: In section 2 we briefly review related work. Afterwards, we present the architecture of our approach for Semantic Web browsing and describe the different stakeholder within the architecture and their interplay. As a proof-of-concept of our approach, we describe our current prototype of a Personal Music Syndicator which facilitates personalized access to podcasts. Finally, a discussion of our approach, and summary and outlook on current work end the paper.

2 Related Work

Currently, we can distinguish two main strategies for creating Semantic Web browsers: The first strategy visualizes RDF documents without taking into account any particularities of the RDF documents. Examples are Piggy Bank, Longwell¹ or Brownsauce². These browsers are, more appropriately, called RDF browsers.

The second strategy for creating Semantic Web browsing is focusing on a certain domain, which might be narrow (as in the case of DynamicView [4] or mSpace [12]) or broad (Haystack [11] or SEAL [5]). These approaches' architectures are all based on a domain-specific fundament requiring considerable modifications for applying them in other domains. At this time there exists no approach that copes with both issues at the same time: Being generic enough to handle any application domain while offering a domain optimized user interface.

Personalizing the access to Web content is a large area of research, which we cannot describe in this paper with sufficient detail. Instead, we want to point out one particular problem that arose in the recent years in the area of adaptive interactive systems: the open corpus problem [3]. The open corpus problem states that today's adaptive systems code require information for the personalization process within the corpus of documents with the effect that the complete corpus of documents must be known (and thus fixed!) during the design time of the system. One very important effect of the open corpus problem is the limited re-usability of adaptive functionality, which might be one of the reasons why personalized systems have not been so prominent in the Web as might have been expected. For the e-Learning domain, we have shown that the separation of personalization-specific information and the document corpus is possible [7]. Based on these results, we started to develop several Web services for the e-Learning domain, which offer encapsulated and re-usable adaptation functionality for e-Learning, and generalized our concept for the realization of Web services which offer personalization functionality in other domains like publication viewing or music recommendation.

¹ <http://simile.mit.edu/longwell/>

² <http://brownsauce.sourceforge.net/>

Finally, related work to our approach can be found in projects which apply Web services in the Semantic Web. Current research here focuses more on enabling technologies like Web services discovery, composition and orchestration (cf. [9, 10]) and less on usability and user-centered visualization.

3 A Service-based Architecture for Realizing Personalized Semantic Web Browsing

Our approach for a Semantic Web browsing offers users a uniform entry point to access the Semantic Web, and in particular to Web services in the Semantic Web. It has been realized as part of the *Personal Reader Framework* [1, 6] which offers an environment for designing, implementing and realizing Web content readers in a service-oriented manner (see Figure 1):

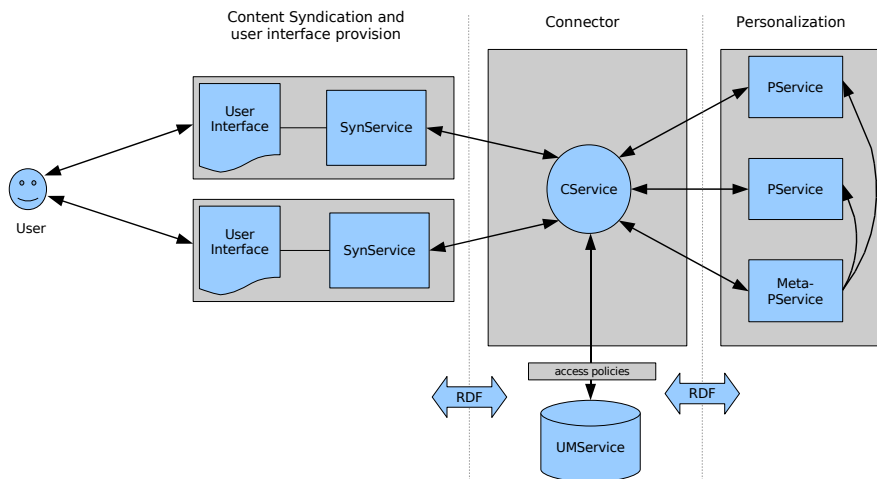


Fig. 1. Overview of Personal Reader architecture

- Web services deliver personalized recommendations for content, extracted and obtained from the Semantic Web. Using Semantic Web techniques they describe their offered functionality in a machine processable format. Optionally, they can return visualization templates that can be used to create user interface snippets for presenting the results of the Web services. We call these kind of Web services *Personalization Services*, *PServices* for short.
- *Meta PServices* can be employed to have single entry points to cooperating or concurrent *PServices*. For example, a music recommender *Meta Personalization Service* orchestrates different *PServices* delivering personalized music recommendations. In an internal processing step the music *Meta-Web service*

filters and orders all the resulting recommendations of the different PServices into one single result.

- For syndicating the results of PServices and for creating appropriate user interfaces, *Syndication Services* (*SynServices* for short), are responsible for displaying the results of several PServices and/or Meta PServices to the user. Each SynService provides such an user end point to a certain domain or task, and allows the user to benefit from many PServices simultaneously, which are selected, combined and customized as the user wishes. SynServices can be realized as RDF browsers, can implement their own RDF processing interface, or they can make use of visualization templates provided by the different PServices / Meta PServices.

The *MyEar Music Syndicator* is an example of such a SynService, which provides the user end point to specify requests for podcasts. Throughout the paper, we use the MyEar Music Syndicator to illustrate our ideas of browsing information in the Semantic Web. Other examples of SynServices provide a user end point for viewing learning objects in an embedding context (where each aspect of this embedding context – like recommendations for quizzes, for examples, for further details with respect to the learning object’s content, etc., is provided by different PServices), or a user end point for browsing scientific publications of a large European project [1].

- For enabling the whole process, a core information provider – a user modeling service, *UMService* for short, deriving appropriate user profiles – is essential. In our architecture, the UMService realizes a centralized approach for user modeling, maintaining and protecting information about a user on behalf of this user. The main reason for choosing a centralized approach is to realize privacy protection: the user has full control about his user profile, and can define policies on which parts of this user profile might be public, or are available to trusted parties only. One central instance for all this information makes it possible to create awareness about stored information, and on realizing policy-protected access. Furthermore, this central UMService receives updates from SynServices or PServices in different domains, and allows for cross-domain re-use of user profile information (if the user wants that).
- From a technical point of view, another component is required to maintain the communication between the SynServices providing the user interface, the PServices, and the UMService. This is the so-called *Connector Service* (*CService* for short) which harvest Web service brokers, collects information about detected PServices (for discovery, selection, customization, and invocation), and for organizing the communication between all involved parties, including requests to the UMService.

3.1 Using the Personal Reader Framework

First, the user has to specify his current request by formulating a query. For the matchmaking between user’s query and provided functionality of detected PServices, the CService offers functionality for the required matching as e.g. described in [8]. The goal is to discover PServices that can incorporate and adapt best to

the provided user data from both, the actual user's request and user profile data. N.B.: Not all PServices need to receive the same user profile data, as some of them might be trusted more than others. The necessary negotiation based on the user-defined policies in the UMService and credentials of the PServices have to be executed beforehand.

The discovery is done as following: First, the Syndicator searches for Web services having the required input and output parameters (*exact match*). This search is send as a request to the CService, which executes the search. If the exact match does not return any results, the CService can do a *plug-in match*, returning Web services that require more input parameters than given. These additional parameters have to be entered by the user in the second step, the configuration step. If the plug-in match cannot find appropriate Web services, *subsumed match*, *subsumed-by match* or a *best-match* can be performed. Which of the matching strategies shall be applied, and which order is preferred, is of course user dependent, the default process will take the order exact – plug-in – subsumed – subsumed-by – best.

Afterwards, all matching PServices will be displayed to the user who can choose which Web service(s) shall be invoked. With this selection step, it is ensured that only Web services are invoked that a user trusts, and negotiations about user profile credentials can be controlled by the user if necessary. Afterwards, PServices' customization parameters – if PServices offer them – are displayed to the user who can adjust them according to his requirements.

Every selected and customized PService is executed and returns its content, plus optionally one or several visualization templates. The visualization templates enable the SynServices to reach a high usability by providing domain-optimized visualization. Default visualization strategies are always available by using a RDF browser. The user can interact with the PServices by clicking on links or completing forms in the generated user interface. As these interactions are sent back to the PService it can adapt it's content more precise to the user's requirements and deliver more personalized content, for example displaying a higher level of detail of the relevant informations.

If a PService detects patterns of usage from the user interaction or certain user requirements, it can send an update request to the UMService. Again, not all PServices are allowed to update the user profile, and the update requests are distinguished according to the overall credentials of a PService, and the credentials of the reasoning process used by a PService to interpret user information. The UMService can allow or deny update requests on behalf of the user. Again, the centralized approach shows its benefits, and approved user profile updates are available immediately to not only the current SynService but to all SynServices for further, user-optimized presentation of Web content.

3.2 Visualization and Interface

All user interaction is realized via the Personal Music Syndicator SynService. It provides the interface for searching and configuring Web services, as described above. After Web services were selected, configured and invoked, the SynService

displays the results of all PServices which have been invoked and returned results. This separation of content collection and syndication / visualization ensures an easy processing of the PServices' output, and it allow the SynServices to adjust visualization according to user devices' capabilities and limitations, or further user preferences.

By delivering visualization templates, every PService can optimize visualization and usability, as certain domain-specific information can be taken into account for creating the user interface.

4 Proof-of-Concept: Personalized access to Podcasts via the Semantic Web

As a proof-of-concept of our approach, we present the *MyEar Syndication Service*, our current prototype of a Personal Music Syndicator, which provides recommendations for music podcast:

Assume a user who searches for podcasts in the Web. He enters a query and receives a list of appropriate podcast delivery services. He specifies which of these services he wants to launch. The user gets a list of all mandatory and optional parameters which can be used to tailor the services – the MyEar Syndication Service tries to fill all these parameters according to the information it has about the user's preferences. The user can change or simply approve these parameters, eventually the user is requested to enter information that the MyEar Syndication Service was not able to provide. Finally, the user gets the syndicated output of all the services he launched, displayed in his personal Web interface. The appropriate visualization is chosen with respect to the currently used display device of the user.

4.1 Current State of Implementation

In our vision, the user can select detected and appropriate PServices which match his query. As we currently lack a sufficient amount of PServices, the procedures which later will execute the matching process between the user request and the PServices is simply returning all available PServices. The user selects which Web services to use, as depicted in Figure 2).

In the second step, the user can configure selected PServices. For example, the MyEar Syndication Service allows the user to specify keywords, duration and iTunes category of the podcasts he wants to listen to. The description of these customization parameters is provided by the PServices. The user profiling which enables the automatic configuration of the PServices is at the moment a simple one: It stores the parameters the user has entered the last time he used this Web service, and returns them as the default selection in the configuration dialog (figure 3).

After configuration, the MyEar Syndication Service is invoked with the specified parameters. This invocation is passed - via the connector - to the corresponding PServices and MyEar receives the determined content (coded as RDF),

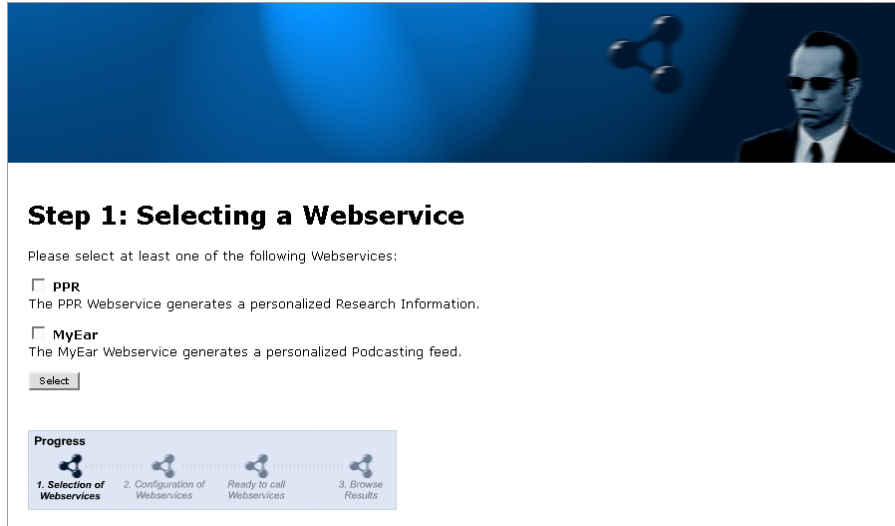


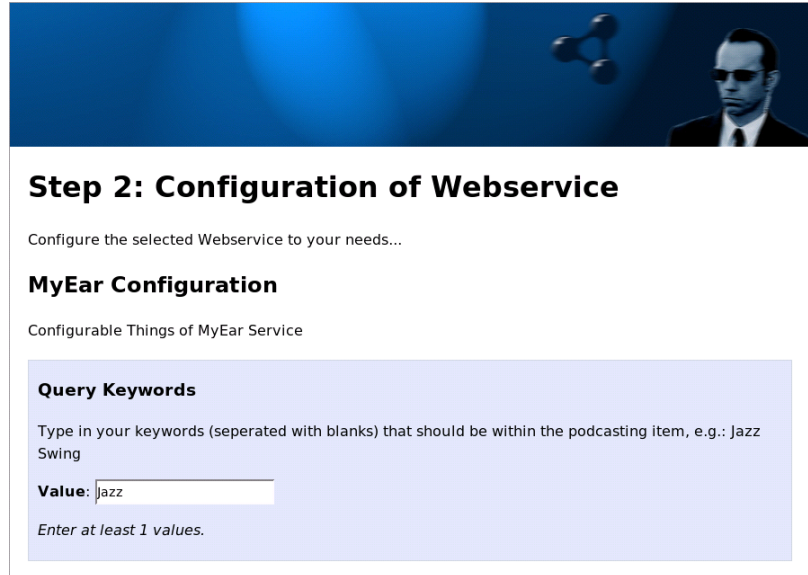
Fig. 2. Dialog for Selecting Personalization Services

as well as visualization templates. Only one visualization template is currently available, which displays the RDF document on PCs within a Web browser, as can be seen in figure 4. The possibility to provide visualization templates by the PServices allows for domain-specific optimization of the user interface, which is not realizable with general-purpose RDF browsing approaches. In the case of the MyEar Syndication Service, for example drag and drop operations are available for selecting podcasts and controlling the audio together with further, music domain-specific gadgets.

4.2 Discussion

A very important issue for improving the usability of Semantic Web browsers is adequate and intuitive visualization of content and its access. A common approach in today's Semantic Web browsers is to visualize raw RDF files while disregarding the domain and purpose of the RDF document. Therefore, usability of such domain independent Semantic Web browsers is weak. Other approaches focus on some specific domain, and may obtain high usability for this single domain. This procedure has the disadvantage that domain spanning-browsing or domain-spanning content processing is not possible. Consequently, the user has to switch browsers whenever he switches between domains. Synergy effects based on a single user interface and domain spanning user profiling cannot be achieved.

Our approach combines domain specific visualization on the one hand, and a single user interface creation and user profile maintenance on the other. There-



The screenshot shows a web interface for configuring a webservice. At the top, there is a blue header with a molecular structure icon and a man's face. Below the header, the main content area is white and contains the following text:

Step 2: Configuration of Webservice

Configure the selected Webservice to your needs...

MyEar Configuration

Configurable Things of MyEar Service

Query Keywords

Type in your keywords (seperated with blanks) that should be within the podcasting item, e.g.: Jazz Swing

Value:

Enter at least 1 values.

Fig. 3. Configuration of the MyEar Syndication Service

fore, we out-source visualization from the Semantic Web browser to the PServices by letting the PServices specify domain-specific and optimized visualization strategies, available via visualization templates. Visualization templates are used by SynServices for creating the final user interface, and – additionally – can optimize the user interface to match the constraints of the currently used device and user specific preferences. To ensure that also results of PServices, which do not provide some visualization template, can be displayed, each SynService implements a generic visualization of RDF documents based on a RDF browser like Piggy Bank³. With this dual approach, a domain-optimized user interface will be provided whenever possible, and, as a fall-back solution, general-purpose RDF visualization is possible. This approach has the advantage that domain modeling has not to be done twice (in the Semantic Web browser and in the PServices); furthermore, changes in the output or the visualization of the PService do not require changes in the Semantic Web browser. The user directly interacts with the visualization from the PService, thus PService-specific interactions are enabled, too.

A further advantage of the architecture of the Personal Readers is that access to the user profile can be restricted to trustworthy instances only. This enables high usability as the same information do not need to be given manually to every single Web service, and privacy protection strategies can be facilitated under the full control of the user. By enabling user profile updates from PService

³ <http://simile.mit.edu/piggy-bank/>

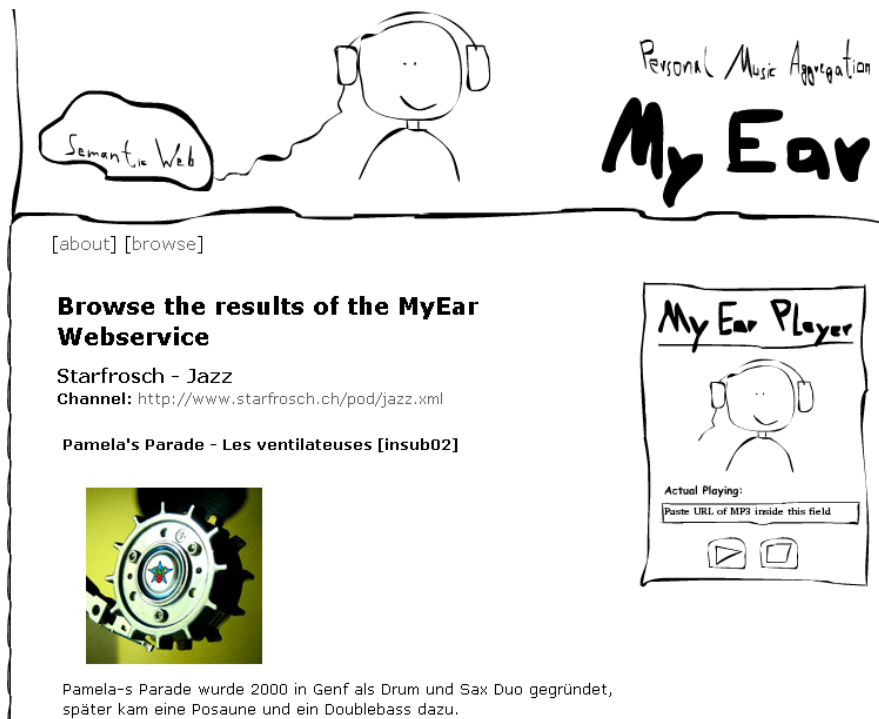


Fig. 4. Visualization of the MyEar Syndication Service

vices, domain-specific user modeling techniques can and should be implemented in the PServices. The very same argument as in the case of visualization also applies here: domain-specific information should be maintained by those parties which naturally have access to it, other approaches always require duplication or at least exchange of domain models which is error-prone, and at least time consuming.

5 Conclusion and Future Work

We have presented our idea for personalized browsing of Semantic Web data, and presented – as one development of the Personal Reader Project – the *MyEar Music Syndication Service*. Our approach to a Semantic Web browser offers domain-spanning and personalized access to Web services in the Semantic Web. We have enhanced Web services to deliver – beside content – customization and visualization information. Our approach enables domain-optimized usability as user interfaces can be designed on a per Web service base. Combined with a generic RDF browser this approach is simultaneously general and domain-

specific, allowing the use of a single user interface and a single user profile to access distributed and personally syndicated Web content.

Our future work will focus on adding more Web services offering personalization for testing the proposed matchmaking strategies within a real world setting.

6 Acknowledgment

We would like the Personal Reader Team at the University of Hannover, and especially Fabian Abel for his engagement in developing the MyEar Syndication Service for the Personal Reader project. This work has been partially supported by the European Network of Excellence "REWERSE - Reasoning on the Web with Rules and Semantics".

References

1. ABEL, F., BAUMGARTNER, R., BROOKS, A., ENZI, C., GOTTLOB, G., HENZE, N., HERZOG, M., KRIESELL, M., NEJDL, W., AND TOMASCHEWSKI, K. The personal publication reader, semantic web challenge 2005. In *4th International Semantic Web Conference* (nov 2005).
2. BERNERS-LEE, T., HENDLER, J., AND LASSILA, O. The Semantic Web. *Scientific American* (May 2001).
3. BRUSILOVSKY, P. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction 11* (2001), 87–110.
4. GAO, Z., QU, Y., ZHAI, Y., AND DENG, J. Dynamicview: Distribution, evolution and visualization of research areas in computer science. In *Proceeding of International Semantic Web Conference* (2005).
5. HARTMANN, J., AND SURE, Y. An infrastructure for scalable, reliable semantic portals. *IEEE Intelligent Systems 19*, 3 (2004), 58–65.
6. HENZE, N., AND KRIESELL, M. 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).
7. HENZE, N., AND NEJDL, W. Adaptation in Open Corpus Hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems 12* (2001).
8. KLUSCH, M., FRIES, B., AND SYCARA, K. Automated semantic web service discovery with owls-mx. In *Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* (2006), ACM Press.
9. MCILRAITH, S., SON, T., AND ZENG, H. Semantic web services. *Intelligent Systems 16*, 2 (2001), 46–52.
10. MOTTA, E., DOMINGUE, J., AND CABRAL, L. Irs-ii: A framework and infrastructure for semantic web services. In *Proceedings of the 2 Intl. Semantic Web Conference* (Florida, USA, 2003).
11. QUAN, D., AND KARGER, D. How to make a semantic web browser. In *Proceedings of the 13th International Conference on World Wide Web* (2004), pp. 255–265.
12. SHADBOLT, N. R., GIBBINS, N., GLASER, H., HARRIS, S., AND SCHRAEFEL, M. C. CS AKTive space or how we stopped worrying and learned to love the semantic web. *IEEE Intelligent Systems 19*, 3 (2004).

Personalisierbare Informationssysteme im Semantic Web

Nicola Henze

ISI – Semantic Web Group, Universität Hannover & Research Center L3S
Appelstr. 4, D-30167 Hannover
henze@l3s.de

Zusammenfassung Dieses Kapitel gibt einen Überblick über die Herausforderungen und Möglichkeiten, die sich durch das Semantic Web für die Personalisierung von Informationssystemen ergeben. Nach einer Definition der Begriffe Personalisierung und Personalisierbare Informationssysteme wird eine kurze Rückschau auf Personalisierungsmethoden in Informationssystemen gegeben. Mit diesem Wissen wird untersucht, welche Ausgangslage für Personalisierung durch das Semantic Web gegeben ist und warum Personalisierung für Anwendungen im Semantic Web wichtig ist. Zwei Beispiele, die Möglichkeiten zur Personalisierung von Informationsangeboten im Web realisieren, beschließen dieses Kapitel.

1 Einführung

Informationen sind ein kostbares Gut: *die richtige Information zum richtigen Zeitpunkt in den richtigen Händen* beeinflusst den unternehmerischen Erfolg ebenso wie die Entscheidungsfindung einer Privatperson. Seit dem Aufkommen des Internet in den 90er Jahren des letzten Jahrhunderts ist der Umgang mit elektronisch gespeicherten Informationen auch aus dem Alltag nicht mehr wegzudenken: Reiseplanung und -buchung im Internet gehören für viele zum Alltag, Produkte lassen sich – nach wenigen Mausklicks – vergleichen, Verträge *online* abschließen, und auch “Electronic Government” ist längst keine Science Fiction mehr.

In all diesen Beispielen ist es immer ein Anwender – ein sogenannter Benutzer¹ – der diese Informationen aufgrund seiner individuellen Anforderungen erfragt, abrufen, vergleicht und auswählt. Deshalb, so sollte man meinen, kommt diesem Benutzer, der den gesamten Prozeß rund um die Informationen initiiert, eine zentrale Rolle bei der Gestaltung von Informationssystemen zu. Jedoch verfolgen die meisten Informationssysteme einen anderen Ansatz: die individuellen Anforderungen des Benutzers werden hintenangestellt

¹ Zur besseren Lesbarkeit wird vom “Benutzer” in der männlichen Form gesprochen, stellvertretend für die Benutzerin oder den Benutzer.

zugunsten von “One-Size-Fits-All” Ansätzen, die jedem Nutzer die gleiche Interaktions- und Steuerungsmöglichkeiten bieten. Im besten Falle werden Formulare, Check-Boxen, Auswahllisten und Ähnliches verwendet, mit denen der Benutzer nun selbst seine Anforderungen im Kontext dieser One-Size-Fits-All-Welt abbilden kann.

Bei der Personalisierung von Informationssystemen stellt man sich daher insbesondere die folgenden Kernfragen:

Wie kann man die Anforderungen der Anwender effektiv modellieren, und wie lässt sich mit diesem Wissen der Umgang mit elektronischen Informationen für den Benutzer effizienter gestalten?

Der Begriff *Personalisierung* meint dabei generell die Anpassung einer Anwendung an die individuellen Anforderungen eines Benutzers. Das kann systemseits geschehen (man spricht dann von *adaptiven* Systemen, die sich selbstständig an den Benutzer anpassen), oder vom Benutzer ausgehen (man spricht dann von *adaptierbaren* Systemen). Beide Ansätze werden in der Praxis oft gemischt, d.h. es liegen adaptive Systeme vor, die der Benutzer selbst aber auch adaptieren kann. Gerade diese Mischform hat sich als vorteilhaft erwiesen, um Kontrollierbarkeit und Transparenz bei der Personalisierung zu erreichen (siehe auch Abschnitt 2.1).

Warum ist nun gerade das Semantic Web eine Chance, personalisierbare Informationssysteme zu gestalten? Dieser Frage wird im vorliegenden Artikel auf den Grund gegangen. Im folgenden Kapitel werden wir uns zunächst genauer mit Personalisierbaren Informationssystemen beschäftigen. Nach diesen Grundlagen untersuchen wir, welche Möglichkeiten das Semantic Web für Personalisierung bietet, und warum Personalisierung, also die effizientere Unterstützung der Endbenutzer beim Umgang mit den Informationen im Web, andersherum einen wichtigen Beitrag im Semantic Web leisten wird (Kapitel 3). Zum Abschluß des Artikels werden zwei Beispiele vorgestellt, die demonstrieren, wie Personalisierung in einem *Semantischen Web* gestaltet werden kann: Am Beispiel des Personal Publication Readers wird gezeigt, wie personalisierte Sichten auf verteilte Web-Informationen realisiert werden können (Kapitel 4.1); die kontextabhängige Zusammenstellung von Web Services ist Gegenstand von Kapitel 4.2.

2 Personalisierbare Informationssysteme

Personalisierbare Informationssysteme, insbesondere Web-basierte Informationssysteme können grob in zwei Klassen unterteilt werden. Zum einen gibt es diejenigen Systeme, die auf dem gesamten Webgraphen arbeiten. In diese Klasse fallen z.B. die sehr erfolgreich im E-Commerce eingesetzten *Recommendersysteme* (vgl. z.B. [13]): Einem Kunden wird ein Produkt empfohlen, da in früheren Verkaufsvorgängen beobachtet wurde, daß Kunden mit ähnlichem

Verhalten wie der aktuelle Kunde sich bereits für dieses Produkt interessierten. Eine Reihe von Algorithmen wurde entwickelt, um die Vorhersagegenauigkeit der Systeme zu optimieren; freie Parameter hierfür sind z.B. das Ähnlichkeitsmaß, das beim Klassifizieren von Benutzergruppen angewendet wird sowie die einzelnen, diskriminierenden Faktoren und ihre Priorität, die Genauigkeit und Aussagekraft der Produkt- oder Vorgangsbeschreibungen, demographische Betrachtungen, Anwendungskontext und vieles mehr.

Komplementär zu diesen Webgraph-basierten-Methoden, gibt es die Klasse derjenigen personalisierbaren Informationssysteme, die auf einem – im Vergleich kleinen – dafür aber um zusätzliche Informationen angereicherten Teilgraphen arbeiten, die sogenannten *Adaptiven Hypermediasysteme* (vgl. z.B. [3]). Adaptive Hypermediasysteme werden vor allem im Bereich E-Learning eingesetzt. Wie der Name besagt, ist bei diesen Systemen der zugrundeliegende Dokumentraum ein Hypertext [9], der sich als Graph auffassen lässt. Zwei verschiedene Adaptionmöglichkeiten werden untersucht: Die Anpassung der Kanten im Graphen (d.h. der Linkstruktur) durch Nützlichkeitsbewertungen der Kanten und daraus resultierend das Hinzufügen oder Löschen von Kanten, deren Bewerten, Sortieren oder Annotieren. Zusätzlich können auch die Knoten selbst (d.h. die Hypertextdokumente) angepaßt werden, in dem geeignete Darstellungen ausgewählt werden, zusätzliche Informationsbausteine hinzugefügt oder nicht-relevante Informationsbausteine entfernt werden.

Adaptive Hypermediasysteme unterscheiden sich von den Webgraph-basierten Systemen, da sie ganz besondere Anforderungen an den zugrundeliegenden Graph stellen: Knoten *und* Kanten in diesem Graph müssen mit zusätzlichen, adaptionrelevanten Informationen angereichert werden, die bei beliebigen Web-Informationen in aller Regel nicht vorliegen. Solche adaptionrelevanten Informationen sind im Bereich des E-Learning z.B. eine intendierte Lernreihenfolge (liegt als Metainformation an den Kanten des Hypertextgraphen) oder die Ausrichtung einer Information an eine bestimmte Zielgruppe (liegt als Metainformation an den Knoten des Hypertextgraphen vor).

Bei adaptiven Hypermediasystemen liegt also ein kleiner, mit adaptionrelevanten Informationen angereicherter Graph vor und in der Regel eine kleinere Zahl von Nutzern, die dann aber mit sehr feinen Modellen abgebildet werden. Die Webgraph-basierten Systeme dagegen werten mit Methoden des Data Mining den Inhalt von Webressourcen aus und analysieren die Verlinkungsstruktur und Nutzung von Webressourcen und -links, um hieraus adaptionrelevante Informationen zu gewinnen.

Für beide Klassen von personalisierbaren Informationssystemen läßt sich sagen, das die Personalisierungsaufgabe als solche eine Optimierungsaufgabe ist: Informationen werden aufgrund der systemseits modellierten Anforderungen eines individuellen Benutzers sowie vorhandener Metainformationen über die Informationen, deren Kontext oder Rolle in (möglicherweise komplexen) Modellen ausgewertet, bewertet und selektiert. Dabei muß der gesamte Prozeß der Informationssuche, -auswahl und -auslieferung berücksichtigt werden,

demzufolge spielt Personalisierung auch während jedes Verarbeitungsschrittes eine Rolle. Im folgenden werden wir zwei mögliche Interaktionsszenarien mit einem Informationssystem genauer analysieren. Diese Szenarien gehen von unterschiedlichen Interaktionsparadigmen aus: Im ersten Fall ist der Benutzer aktiv und fragt Informationen ab (Pull-Szenario), im zweiten Beispiel ist das System aktiv und informiert den Anwender selbsttätig über Informationen (Push-Szenario).

Benutzerinteraktionen mit einem Informationssystem: Pull-Szenario

In diesem Szenario gehen wir davon aus, daß ein Benutzer einen Informationsbedarf hat, zur Vereinfachung gehen wir davon aus, dass dieser Informationsbedarf als Informationsanfrage vorliegt. Diese Informationsanfrage wird nun zunächst in eine – in Bezug auf das Informationssystem – gültige Anfrage übersetzt. Bei diesem Übersetzungsschritt können individuelle Anforderungen des Benutzers berücksichtigt werden, z.B. Anwendungskontext (der Benutzer ist z.B. gerade in einer Kundenberatung, oder er informiert sich generell über die Neuerungen in der Firma), oder Präferenzen für bestimmte Informationsanbieter, etc. Das heißt, in diesem ersten Informationssucheschritt kann die Anfrage aufgrund von Informationen aus einem Benutzermodell verfeinert werden. Im anschließenden Retrieval-Schritt kann eine zusätzliche, individualisierte Bewertungsfunktion die Genauigkeit der Reihung der gefundenen Informationen überarbeiten. Die Auswahl der Informationen erfolgt nun, indem z.B. Kosten kalkuliert und Qualität evaluiert werden und mit dem vom Benutzer tolerierten Werten verglichen werden, ggf. Einschränkungen des vom Benutzer gerade verwendeten Endgeräts (PC, PDA, Handy, etc.) berücksichtigt werden, etc. Nachdem nun alles für die Auslieferung vorbereitet ist, ist noch die individualisierte Präsentation der Ergebnisse zu erledigen.

Benutzerinteraktionen mit einem Informationssystem: Push-Szenario

Das zweite Szenario, das wir hier noch kurz betrachten möchten, ist durch ein aktives System und einen passiven Benutzer gekennzeichnet. Das System lernt einen Filter, der den Informationsbedarf des Benutzers beschreibt. Schritte und Personalisierungsmethoden, die im Pull-Szenario angesprochen wurden, gelten hier entsprechend, um diesen Filter zu bestimmen. Das System arbeitet dann im Batch Modus und sucht kontinuierlich nach neuen Informationen, die gemäß dieses Filters den Benutzer interessieren können. Als Beispiel sei ein “News-Watch System” genannt, das ständig die neuesten Nachrichten aufgrund ihrer Relevanz für den Anwender bewertet. Ein Filter kann z.B. alle Nachrichten, die über den Reuters Newsticker eingehen und innerhalb von 30 Minuten Einzug auf mindestens einer renommierte Internet-Nachrichtenseite finden, pro-aktiv im Interface des Benutzers anzeigen.

2.1 Transparenz des Personalisierungsprozesses

Sehr wichtig ist es, die Personalisierungsaktionen für den Benutzer transparent zu gestalten, und dem Benutzer die Kontrolle über jeden der einzelnen Schritte so einfach wie möglich zu machen. Dies ist keineswegs trivial, aber nichtsdestotrotz essentiell, wenn es um die Benutzerakzeptanz und damit letztlich um den Erfolg des Systems geht. Folgendes Beispiel mag das illustrieren: Eine Information, zertifiziert durch eine anerkannte Institution und damit als qualitativ hochwertig einzuschätzen, wird herausgefiltert, da der Preis für diese Information den vom Benutzer eingestellten Toleranzwert überschreitet. Auch wenn der Benutzer in vielen Fällen gesuchte Informationen kostenlos oder für einen Preis im Bereich seines Toleranzwertes findet, wird es sehr wahrscheinlich Situationen geben, in der nur ungenügend Information gefunden werden und die beste Antwort aber – aufgrund der Filterkriterien – zunächst herausgefiltert wird. Meldungen wie “Zertifizierte Informationen zu Ihrer Anfrage sind für einen Preis P zu erhalten” geben dem Benutzer die Möglichkeit, mit einfachen Mitteln die Personalisierung von Fall zu Fall modifizieren zu können, ohne insgesamt auf die Unterstützung durch die Personalisierung verzichten zu müssen.

2.2 Diskussion

In diesem Kapitel wurde dargestellt, dass Methoden zur Personalisierung von Informationssystemen dahingehend unterschieden werden können, auf welcher Grundmenge sie arbeiten: Auf mit Metadaten angereicherten Graphen (Adaptive Hypermedia) oder auf Webressourcen im Webgraphen (Recommendersysteme).

Das Semantic Web mit seinem formalen Ansatz zur Annotation von Webinhalten mit maschinenlesbarer Bedeutung liefert hervorragende Ausgangsvoraussetzungen, um die adaptionsrelevanten Informationen, die im Adaptive Hypermedia verwendet werden, zu modellieren. Jedoch wird der Ansatz, das Semantic Web einzig als Technologielieferant für Metadaten im Hypermediagraphen zu verwenden, dem Gesamtkonzept Semantic Web nicht gerecht und vernachlässigt die Möglichkeiten zur dynamischen Generierung von Kontexten und damit letztendlich auch zur dynamischen Generierung von adaptionsrelevanten Informationen.

Im Gegensatz dazu bieten die Webgraph-basierten Methoden, die sich im wesentlichen auf Miningverfahren und statistische Analysen stützen, im Moment keine hinreichenden Ansätze, um mittels formaler Schlußfolgerungen über die *Bedeutung* von Inhalten neue Zusammenhänge aufzudecken.

Beide Verfahrenstypen bringen erfolgreich evaluierte Ansätze zur Personalisierung mit, und beide Verfahrenstypen können vom Semantic Web profitieren. Im nächsten Kapitel werden wir sehen, warum das Semantic Web Personalisierung fördert – und sogar fordert.

3 Die Herausforderung des Semantic Web

“The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.” Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001 [2]

Die Entwicklung des *Semantic Web* zielt darauf ab, die Bedeutung von Informationen im World Wide Web maschinenlesbar und vor allen Dingen *maschinenverarbeitbar* darzustellen, so daß Applikationen den Benutzer im Umgang mit dem Web effizienter als bisher unterstützen können. Insbesondere der zweite Teil dieses Zitats, “people” – also Benutzer – besser beim Umgang mit Informationen im Web zu unterstützen, weist noch einmal darauf hin, daß es die Benutzer sind, die von den entwickelten Technologien profitieren sollen.

Um das Semantic Web Realität werden zu lassen, braucht man Formalismen (wie z.B. RDF [10]), Sprachen (wie z.B. RDFS [11] und OWL [8]), Anfrage-, Regel- und Schlußfolgerungssprachen (wie z.B. RDQL[12], SWRL[16] oder WRL [17]), etc., die zur Zeit entwickelt werden. Auch wenn es heute noch hauptsächlich die Methoden und Technologien sind, die die Hauptaufmerksamkeit genießen, bleibt festzustellen: der Effekt all der Entwicklungen wird erst in den Anwendungen deutlich, die Nützlichkeit eines Semantic Web kann erst durch verbesserte Applikationen – und damit schlußendlich durch den Endanwender – nachgewiesen werden.

Die Bedeutung des Semantic Web für Personalisierung und personalisierbare Informationssysteme liegt in der Hauptsache darin, daß Webressourcen mit maschinenverarbeitbarer Bedeutung versehen werden. Diesen Sachverhalt können wir auch anders ausdrücken: Inhalt wird von der eigentlichen Auslieferung bzw. Präsentation getrennt beschrieben. Warum ist dies so? Wenn Inhalte *und* ihre Bedeutung formal und für Computerprogramme aufbereitet beschrieben werden, heißt das, das Inhalte in verschiedenen Kontexten verwendet werden können – so weit eben das, selbstverständlich beschränkte “Verständnis” von den Inhalten, das ein Computerprogramm durch die formalen Beschreibungen und Wissensbasen erhält, dies zuläßt. Ein Beispiel: Das “Wissen”, daß eine Zeichenfolge die Homepage einer Forschungseinrichtung präsentiert, wobei diese Forschungseinrichtung (bzw. dessen abstrakte Repräsentation) in der Relation “Kernkompetenzen” mit einer Menge von weiteren Begriffen steht, die ihrerseits Referenzen auf Einträge in einer oder mehrerer Wissensbasen / Ontologien sind, kann auf unterschiedliche Arten genutzt werden. Es können z.B. Anfragen nach der größten gemeinsamen Menge von Kernkompetenzen von Forschungseinrichtungen in einem bestimmten Land unterstützt werden. Hierbei muß weder der Begriff “Forschungseinrichtung” noch die Relation “Kernkompetenz” literal verwendet werden: die Verbindung zwischen Ontologien, das sogenannte *Ontologymapping* und *Ontologymerging*, bereiten die Möglichkeit, durch das Auflösen auf Bedeutungsebene unterschiedliche Bezeichner aufzudecken. Auch die Verortung einer Forschungseinrichtung in einem bestimmten Land kann ermittelt werden, z.B.

indem die Adressangaben (Stadt) unter Zuhilfenahme einer geographischen Wissensbasis über Länder, Städte, etc. ausgewertet wird.

Die Quintessenz hiervon ist, das Inhalte immer wieder neu zusammengestellt werden können, immer wieder neue Sichten auf Inhalte möglich werden, und diese Sichten sogar – durch das Schlußfolgern auf den Ontologien und Wissensbasen – neue Zusammenstellungen, neue Relationen oder neue Kontexte sein können. Und es liegt nahe, diese Sichten auf die Inhalte nicht nur formal gemäß Wissensbasen und ggf. Regelmengen zu erstellen, sondern bei diesem gesamten Informationsbeschaffungsprozeß (siehe Kapitel 2) die Benutzeranforderungen zu berücksichtigen, d.h. personalisierte Sichten auf Inhalte des Web zu ermöglichen.

4 Beispiele

Dieses Kapitel führt zwei Beispiele für Personalisierung im Semantic Web auf: Der Personal Publication Reader (siehe Abschnitt 4.1) demonstriert die Zusammenfassung von Informationen aus verteilten, sich in Format, Layout und Struktur unterscheidenden Webseiten in einer integrierten, personalisierten Darstellung. Das zweite Beispiel (siehe Abschnitt 4.2) zeigt auf, wie der Kontext eines Benutzers, z.B. Endgerät oder Ortsinformationen, genutzt werden können, um Web Services personalisiert anzubieten.

4.1 Personalisierte Sichten auf verteilte Inhalte

Der Personal Publication Reader² zeigt, wie in einem *Semantic Web* Informationen von verschiedenen Quellen zusammengefaßt und vermittels Personalisierungsregeln angezeigt werden können [1]. Der Personal Publication Reader löst dieses Problem für das folgende Szenario (in Anlehnung an [1]):

Peter arbeitet als Forscher an einer Universität. Er veröffentlicht seine Forschungsergebnisse in wissenschaftlichen Zeitschriften und auf Konferenzen. Außerdem stellt er – wenn möglich – seine Publikationen auf seiner Homepage online zur Verfügung. Peter arbeitet außerdem in einem internationalen Forschungsprojekt mit. In regelmäßigen Abständen wird er aufgefordert, seine aktuellen Publikationen an das Projektmanagementbüro zu übermitteln. Das Projektmanagementbüro unterhält eine Webseite mit Informationen über die Mitglieder des Projektes, ihre Aufgaben im Projekt, etc. und natürlich die Publikationen des Projekts.

Bei der Analyse dieses Szenarios fallen zwei grundlegende Aspekte auf:

² www.personal-reader.de

1. Die Daten werden dupliziert: auf der Homepage des Wissenschaftlers, auf der Homepage des Projekts. Die Duplikation von Daten ist jedoch kritisch, da fehleranfällig und aufwendig in der Pflege.
2. Die Informationen, die regelmäßig an das Projektmanagementbüro übertragen werden, sind bereits online vorhanden – aber nicht für weitere Verarbeitung verfügbar: Projekt, Personen, Publikationen, Beiträge der Partner, Forschungsprofile, ...

Der Personal Publication Reader bietet die folgende Lösung an: In einem ersten Schritt werden die Daten über die Wissenschaftlichen Veröffentlichungen von der Webseite der Forschungseinrichtungen oder Unternehmen, die an einem gemeinsamen Forschungsprojekt teilnehmen, extrahiert. Dies geschieht unter der Verwendung des Toolkits von Lixto³, das die semi-automatische Extraktion von Webdaten unterstützt. Lixto stellt ein visuelles, interaktives Programm zur Erstellung von sogenannten *Wrappern* zur Verfügung, die zur Erkennung der zu extrahierenden Daten von einer Webseite genutzt werden [5]. Darüberhinaus wird die regelmäßige oder on-demand Extraktion von Informationen und deren Transformation in die Sprache XML [18] unterstützt. Der Personal Publication Reader verwendet diese Methoden zur Webdateneextraktion, um – nach geeigneten Transformation in RDF – eine maschinenlesbare Beschreibung der wissenschaftlichen Veröffentlichungen der Projektteilnehmer zu generieren, die sich des Vokabulars des Dublin Core, eines für Bibliotheken entwickelten Standards zur Beschreibung von Veröffentlichungen [4], bedient. Am Ende dieses ersten Schritts liegt also eine Beschreibung der Wissenschaftlichen Publikationen aller Partner in einer Sprache des Semantic Webs vor.

In einem zweiten Schritt wird eine Ontologie konstruiert, die das Projekt selbst beschreibt. Diese Ontologie erweitert eine bestehende Ontologie, die Semantic Web Researcher Community Ontology [15], um projektspezifische Aspekte wie Arbeitsgruppen, die Rolle von Personen in einem Projekt, etc. Nun stehen zwei unterschiedlichen Quellen mit unterschiedlicher semantischer Tiefe zur Verfügung: Die Informationen über wissenschaftliche Veröffentlichungen in RDF (unter Verwendung eines kontrollierten Vokabulars, des Dublin Core Standards), sowie die Projektontologie in OWL, die u.a. Beziehungen zwischen Personen, Arbeitsgruppen, Forschungszielen, etc. modelliert. Regeln erlauben nun Schlußfolgerungen auf diesen Daten: Z.B. sind eine Reihe von Veröffentlichungen relevant für eine gegebene Publikation, wenn die Autoren der letzteren mit den Autoren der ersteren in einer gemeinsamen Arbeitsgruppe im Projekt arbeiten. Oder: Für Benutzer mit einem bestimmten Forschungsinteresse können Veröffentlichungen vorgeschlagen werden, wenn die Autoren dieser Veröffentlichungen ähnliche Interessen wie der Benutzer haben. Im Personal Publication Reader werden diese Regeln *Personalisierungsregeln* genannt.

³ www.lixt.com

In einem letzten Schritt wird nun noch das Benutzerinterface erzeugt. Die Personalisierungsregeln werden im Personal Publication Reader als Web Services zur Verfügung gestellt. Diese Web Services sind in der Lage, neue, personalisierte Sichten auf die ursprünglich verteilt im Web vorliegenden Daten zu generieren, indem sie diese Personalisierungsregeln anwenden. Die Sichten – und damit das Ergebnis des Reasoning in den Web Services – werden wiederum in RDF beschrieben und durch eine Reihe von weiteren Komponenten in einem Benutzerinterface visualisiert [6].

4.2 Personalisierung von Web Services: Kontext-sensitive Dienste

Eine andere Art, Personalisierung im Semantic Web zu realisieren, wird in [7] beschrieben. Wie im Beispiel des Personal Publication Readers geht es auch hier um eine Web Service-orientierte Architektur, doch liegt der Fokus nicht auf die Implementierung der Web Services selbst, sondern um den Aufruf und die Zusammenstellung von bereits vorhandenen Web Services.

Ein Szenario, daß die Nützlichkeit der personalisierten Zusammenstellung von Web Services illustriert, ist das folgende (in Anlehnung an [7]):

Ein Restaurantführer empfiehlt seinen Kunden Restaurants in einer Stadt. Da die Kunden ihre Entscheidung für ein Restaurant oftmals davon abhängig machen, wie gut sie das Restaurant erreichen können, bietet der Restaurantführer auch gleich einen Wegeplaner-Service an, der – abhängig vom momentanen Aufenthaltsort der Kunden – eine Wegbeschreibung zum Restaurant liefert.

Aufgrund des Erfolgs des Restaurantführers plant das Unternehmen, nun auch einen Hotelführer in gleicher Art zur Verfügung zu stellen. Wie kann der Hotelführer den Wegeplaner-Service des Restaurantführers nutzen?

In diesem Szenario ist es der Kontext des Benutzers – hier der Aufenthaltsort – der weitere Aktionen hervorruft: das Aufrufen des Wegeplaners. Eine solche Komponente wie der Wegeplaner sollte von verschiedenen Services wie z.B. dem Restaurantservice oder dem Hotelservice gemeinsam benutzt werden können, um eine kontextabhängige Dienstleistung als Erweiterung der eigentlichen Servicefunktionalität zur Verfügung zu stellen.

Um die (Wieder-) Verwendung solcher kontextabhängigen Web Services zu vereinfachen, wird in [7] eine Erweiterung des zur Zeit am häufigsten verwendeten Protokolls zum Aufruf von Web Services, des SOAP Standards, [14] vorgeschlagen: Im sog. SOAP-Header, also dem Bereich, der Verarbeitungs- und Adressateninformationen etc. enthält, wird ein neues Feld für Kontextparameter eingeführt. Damit können Kontextparameter wie z.B. Ort und Zeit in einer SOAP Nachricht in einem gesonderten Bereich übermittelt werden und stehen allen Beteiligten, die diese SOAP Nachricht bearbeiten, direkt zur Verfügung. Die Verwendung und Bearbeitung der Kontextparame-

ter kann dann direkt an *kontextsensitive* Web Services wie den Wegeplaner-Service weitergeleitet werden, die ihren Service nun unabhängig vom eigentlichen Adressaten der Nachricht anbieten können. Mit diesem Vorgehen wird die Wiederverwendung der Web Services in verschiedenen Anwendungen unterstützt.

Für die Benutzer heißt dies, das ihnen – in Abhängigkeit von ihrem momentanen Aufenthaltsort und Kontext – gleiche Dienstleistungen in den unterschiedlichsten Anwendungsszenarien angeboten werden können.

5 Zusammenfassung

Durch das Semantic Web und die Einführung von formal und maschinenverarbeitbar beschriebener Bedeutung von Web Informationen ergeben sich interessante und neue Perspektiven für die Personalisierung von Webinteraktionen, insbesondere für die personalisierbare Interaktion mit Web-basierten Informationssystemen. Die explizite, verwendungsunabhängige Beschreibung von Webinformationen ermöglicht neue, integrierte Sichten auf Informationen, neue Kontexte und Anwendungsszenarien, sowie neuartige Anfragen, die das Schlußfolgern über Informationen erfordern. Durch Personalisierung können diese Informationen für einen Benutzer (oder eine Benutzergruppe) optimiert ausgewählt, zusammengestellt und angezeigt werden. Statt eines “One-size-fits-all”-Ansatzes rückt nun der Benutzer und seine Interessen in den Mittelpunkt der Anwendung: Im Idealfall sollte ein personalisierbares System für den Benutzer so erscheinen, als sei es speziell für ihn und seine Anforderungen entwickelt. In diesem Kapitel sind, ausgehend von einem kurzen Review der grundlegenden Techniken personalisierbare Informationssysteme, die Möglichkeiten und Herausforderungen diskutiert worden, die sich für Personalisierung im Semantic Web ergeben. Zwei Beispiele demonstrieren, wie die Realisierung dieser Personalisierung im Semantic Web gestaltet werden kann.

Literatur

1. R. Baumgartner, N. Henze, and M. 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.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
3. P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87–129, 1996.
4. Dublin Core, 2004. <http://dublincore.org/>.
5. G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, and S. Flesca. The Lixto Data Extraction Project — Back and Forth between Theorie and Practice. In *ACM Symposium on Principles of Database Systems (PODS)*, volume 23. ACM, June 2004.

6. N. Henze and M. 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.
7. M. Keidl and A. Kemper. Towards context-aware adaptable web services. In *Proceedings of the 13. WWW Conference*, volume 13, New York, May 2004. ACM.
8. OWL, Web Ontology Language, W3C Recommendation, Feb. 2004. <http://www.w3.org/TR/owl-ref/>.
9. R. Rada. *Interactive Media*. Springer, 1995.
10. Resource Description Framework: Concepts and Abstract Syntax, W3C Recommendation, Feb. 2004. <http://www.w3.org/TR/rdf-concepts/>.
11. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, Feb. 2004. <http://www.w3.org/TR/rdf-schema/>.
12. RDQL - query language for RDF, Jena - Semantic Web Framework, HP, 2005. <http://jena.sourceforge.net/RDQL/>.
13. U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In *Proceedings of CHI'95 - Human Factors in Computing Systems*, pages 210-217, 1995.
14. SOAP: Simple Object Access Protocole, June 2003. <http://www.w3.org/TR/soap/>.
15. SWRC - Semantic Web Research Community Ontology, 2001. <http://ontobroker.semanticweb.org/ontos/swrc.html>.
16. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Submission, May 2004. <http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/>.
17. WRL: Web Rule Language, Sept. 2005. <http://www.w3.org/Submission/WRL/>.
18. XML: eXtensible Markup Language, W3C Recommendation, Feb. 2004. <http://www.w3.org/XML/>.

Implementation of a Personalized Assessment Web Service

Lilia Cheniti-Belcadhi¹, Nicola Henze², Rafik Braham¹

¹PRINCE Research Group, ISITC, University of Sousse, Tunisia

²IVS-Semantic Web Group, University of Hannover, Germany
cheniti@l3s.de, henze@l3s.de

Abstract

This paper describes the design, development and qualitative evaluation of a web-based personalized assessment service of an object-oriented programming course at the University of Sousse. The assessment service was integrated in the Personal Reader software, of the University of Hannover and was designed to select specific questions based on the progress and performance in the course of each learner individually. This paper identifies benefits to learners brought about with the use of a personalized assessment service. It describes and discusses a first experiment of this tool. The evaluation was completed with the assistance of a group of ten students with different levels of prior programming knowledge. The students are enrolled in a computer science degree program at the Informatics College (ISITC) of the University of Sousse. The personalized assessment framework is introduced, with a description of its architecture and functionalities. An analysis of learner's feedback following this experiment, is also provided.

1. Introduction

Assessment is an essential part of web-based learning as it provides a measure of the outcome of knowledge provided. The design of assessment tools in open environments and over distributed repositories demands effective personalization approaches which provide learners with guidance and individualized support [2]. It also requires advanced approaches to assessment which are appropriate to distributed and open e-learning environments. In those environments, indeed, personalization plays an important role [3] as the information derived from both assessment functionalities and learners' interaction with the environment enables a better selection and generation of learning and assessment strategies. This personalization approach in education contexts should

also enable a better understanding of the learner and tasks that are critical to the learning process. Several web-based assessment systems, such as PASS [5], SIETTE [6], and QuestionMark [11], have already been developed and deployed. Some of these are concerned with the adaptability aspects in the provision of assessment in e-learning environment. Others have rather taken into consideration the conformity to assessment standards such as IMS/QTI [7]. These systems lack sufficient support for generating personalized assessment over heterogeneous learning resources, while conforming to specific standards. This paper describes a personalized assessment framework applicable to open distance learning systems. It is based on a dynamic processing and networking of heterogeneous resources, potentially authored by different people with different goals.

In this framework a personalized assessment web service for open e-Learning environments is proposed, one that draws on recent research in the Semantic Web [1] in which machines can understand and process heterogeneous resources. A first experiment is presented and described, in which the use of the personalized assessment Web Service as part of the Personal Reader system [10] is examined. A group of ten learners used this framework while they took part in a course on object-oriented programming in C++. Their progress and responses to this new methodology of assessment were recorded in a series of questionnaires. The remainder of this paper is structured as follows: In section 2 the architecture of the system and its functionalities are described. In Section 3 an evaluation experiment is illustrated. The experimental settings and established hypotheses are first detailed. A discussion of the results of this evaluation, followed by concluding notes and a statement of inferences for future work are suggested in Section 4.

2. Personalized Assessment Framework

Let us start with a specific scenario, involving a learner, Alice, a computer sciences student, who is preparing for an exam on object-oriented programming. Alice would like, for instance, to review a lesson on the main concepts of objects-oriented programming. The personalized assessment framework evaluates that Alice requires some prerequisite knowledge. Questions will be searched and selected to Alice. These questions constitute the “Pre-assessment test” on the selected lesson. Based on the results of the test conducted by Alice, chapters of the lesson that still await pre-assessment will be highlighted. In these parts, there are some questions to which Alice has not responded. Finally, to ensure that Alice understands the important concepts of the selected lesson, questions will be searched and presented in the form of a personalized post-assessment test on these concepts. Parts of the lesson for which a post-test has been created will also be highlighted.

The main goals of the personalized assessment system are:

- Keeping track of the learner’s progress in revising the course.
- Providing the learner with the appropriate assessment resources related to his/her progress after checking the saved performance, language preferences and device requirements.
- Providing the learner with an accurate estimation of his/her knowledge by creating personalized pre-tests and post-tests
- Providing the learner with a generic assessment framework that can be used in case a detailed description of the course is given.

2.1. Architecture

The schematic representation proposed for the personalized assessment framework is illustrated in Fig.1. In this system, personalized web services are offered, which deliver assessment and generate learning resources with respect to domain ontology, learners’ requirements and interaction with the learning environment. The proposed generic assessment framework entails a dynamic binding amongst its components, notably through user interface services and mediation facilities between learner requests and available personalization goals. Communication between the different Web services is carried out through RDF documents. The RDF descriptions [13] refer to ontologies that permit the search and presentation of the information needed.

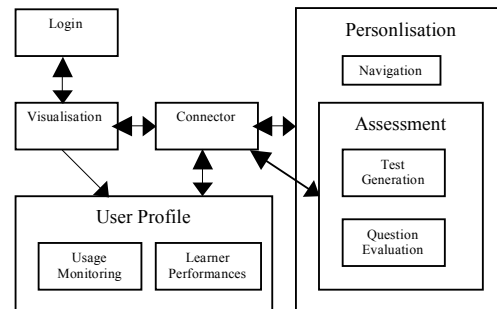


Fig.1. Architecture of assessment framework

To enable personalized assessment support as described in the scenario above, meta-information about the course, the learning and assessment resources and about the learners themselves is required. The Assessment framework makes use of learning standards LOM [9] and Dublin Core [4] to support interoperability. It is also designed to be IMS-QTI compliant, to facilitate the exchange of assessment resources. The annotation of the learning content according to e-learning standards facilitates its reuse and personalization to various learners.

The authentication of learners is completed through the login service, which checks the learner’s parameters and transfers them to the other services. The *visualization service* is responsible for the display of the resources requested by the learner, i.e. either the learning resources or the assessment resources. The communications between all services, except for the login, go through the *connector service*.

The assessment framework is also comprised of the following elements:

- A *navigation service* that provides a personalized navigation through the learning and assessment resources on the basis of the information provided in the learner’s profile.
- An *assessment service* composed of two other services that can be requested independently: *the test generation service* and *the question evaluation service*. Its role is to provide a personalized assessment to the learner based on the information provided in his/her profile. The test generation service is responsible for the construction of tests. A test is typically a well-organized sequence of questions selected on the basis of the profile provided by the connector service. Two types of tests can be generated: pre-test and post-tests. Several types of question can be deployed in a test to evaluate the level of knowledge acquisition by the learner. The question evaluation service is in charge of assessing the learner’s answers.
- A *user profile service* responsible for the generation of a learner’s profile featuring his/her history and built upon the last interactions saved by the learner and answers provided to the proposed tests. It is also

composed of two services: *the usage monitoring service* and *the learner performance service*. The learner's interactions with the assessment and learning resources are recorded through the usage monitoring service. The performance service updates the information on the learner based on his results in the pre- and post- tests. The framework is designed in an open architecture that makes it compatible with open standards and protocols and adaptable to different implementation scenarios.

2.2. Personalized Assessment Framework Functionalities

The framework provides two options for assessment: (i) pre-test, (ii) and post-test. These are performed on the ground of the learning resource that the learner has already selected, thus making it possible to get an idea on his/her performance at different stages of the learning process. The adequate exploitation of this information can enhance the personalization of the learning and the assessment resources provided to the learner.

- Pre-test functionality: Pre-test has been designed to investigate the learner's prior knowledge of selected learning resource and initialize the learner's profile accordingly.

- Post-test functionality: Post-test functionality aims to stimulate the learner to contemplate and reflect on the subject studied, assess his/her performance and evaluate the learning outcomes.

The construction of Pre-tests and Post-tests is a dynamic process that depends on the learner's current level of knowledge. Furthermore, the navigational behaviour of the learner through the content is considered. The assessment procedure takes into account the learning resources that the learner has visited. This type of information is saved in the learner's profile. In the Pre-tests, questions relating to the prerequisite concepts of the selected learning resources are posed. In Post-tests, questions that are relevant to each objective of the current selected resource are presented to the learner. The assessment system allows every learner who uses it to have a personalized interface, which they can access by logging in with their password. Fig.2 is a screen shot showing how learning resources are presented to the learner. In the right part the specific learning resource is displayed. The left part visualizes the results received via the Connector Service. Learning resources for which a pre-test can be constructed based on the information in the learner's profile will be marked with a yellow ball. In case a post-test can be created for the selected learning resource a red ball will be displayed.

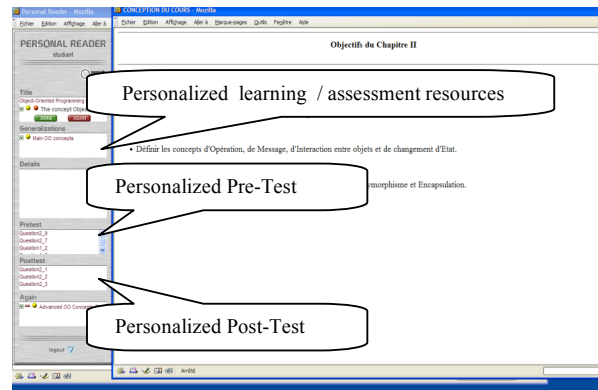


Fig.2. Personalized learning resource

In case an assessment resource is selected, it will be displayed on the window. The left window visualizes the results received via the Connector Service. The possible answers' choices are contained in the details section.

Once a choice is selected by the learner, it will be recorded in the profile, and the corresponding feedback (whether the response is correct or not) will then be displayed in the right part. In case the assessment resource has been answered correctly by the learner, it will be saved as solved in the profile and will not be associated anymore to a Pre-test or Post-test of a learning resource. The personalized assessment approach adopted in our framework works as follows:

- **1st step:** The framework looks for the initial level of knowledge by requesting the learner's profile from the User profile Service.

- **2nd step:** Once the learner selects a learning resource, metadata of the currently visited learning resource will be identified. The connector service submits to the assessment service the description of the learning resource, together with the ontology and the learner's profile. The assessment service selects then the assessment resources that have to be associated to the Pre-and Post-Tests of the selected learning resource. It possesses reasoning rules which makes it able to query for assessment resources and metadata, and reason over distributed data and metadata descriptions. In implementing the reasoning rules [8], the query and rule language for the Semantic Web TRIPLE [12] has been used. Rules defined in TRIPLE can reason about RDF-annotated information resources.

- **3rd step:** the results of personalized pre-tests and post-tests generation together with the personalized assessment recommendation for the selected learning resource (red and yellow balls) will be displayed to the learner thanks to the visualization service.

- **4th step:** Once the learner selects an assessment resource, its metadata will be identified. When he

works on the assessment resource, the answers will be evaluated through the question evaluation service, and will be forwarded to the user profile service and saved in the profile. Assessment resources that have been correctly solved by the learner won't be considered anymore as candidate assessment resources.

3. Evaluation

In this section we describe the development and initial evaluation of the first version of our assessment system. It is important to ensure that the system is empirically operational and academically useful. To this end, a qualitative evaluation was performed on the system, with a particular focus on usability and effectiveness aspects. Two categories of learners: *beginners* and *advanced* has been defined. A panel of ten students with different a priori knowledge of C++ programming was selected. Five of these were enrolled in the second year of a computer science bachelor's curriculum, and were considered part of the "*beginners*" category. The remaining students were in their first year of computer engineering studies at the Informatics College at the University of Sousse. They were gathered in our experiment under the "*advanced*" rubric, as they had taken already a year course in object-oriented programming. In this evaluation each student – without undergoing any prior training on how to operate the system- was given 30 minutes to navigate through the course content and answer some of the 24 questions on object-oriented programming with C++.

3.1. Experiment Hypotheses

A questionnaire was designed to evaluate opinions about this assessment framework and was distributed to all learners. The questionnaire was composed of 19 questions. They all completed and returned the questionnaire. The answers' choices for the first eighteen questions ranged from 1 (strongly disagree) to 5 (strongly agree). The last question was free text question. The learners were observed during this experiment by a lecturer on the course. Verbal communication was initiated to get instant feedback from the learners while they were using the system. The questions were designed, to check the correctness of hypothesis, which we enunciated while conducting the experiments (learner verbal communication and questionnaire). These hypotheses are:

-H1: The system is simple and intuitive to use, with a minimum amount of explanation.

-H2: The design of a Pre-test positively affects the personalization of the learning and the system appropriation.

-H3: The design of a Post-test positively affects the personalization of the learning and the system appropriation.

-H4: The assessment personalization positively affects the learner motivation.

-H5: The system can be easily used for other courses.

-H6: The tracking of the learner performance and progress and the instant feedback provided by the system positively affect the learning process.

-H7: The system provides guidance and orientation to the learner, which facilitates navigation through the resources.

The questionnaire was composed of six questions related to H1, two questions on each of H2, H3, H4 and H6, one question on H5 and three questions on H7. Besides hypothesis testing, the aim of the evaluation was to gain information for further developments of the assessment service and the Personal Reader System.

3.2. Experiment results and discussion

We first present the numerical results (Fig.3) with a focus on our hypotheses validation requirements. As mentioned before, the learners were supervised by a tutor of this course during the test. Fig. 3 shows the average of the questions answers related to each hypothesis: e.g. regarding H1 the average of learners answers to the six questions was 4.23 for advanced learners and 4.5 for beginners. The system was evaluated as easy to use and learn, in addition to being potentially capable of offering higher levels of personalization as traditional computer-based training. The tutor's observations confirm those findings. The analysis of the diagram below and comparison of results for hypothesis H2 and H3 reveals that the learners were more receptive to the personalized post-tests than the pre-tests. This perception also emerged in our discussions with the learners. They have indeed confirmed that the pretest may put them off, and suggested, that the number of questions included in post-tests should be larger than thus included in pretests. Another interesting result relates to H4: we see indeed that beginner learners felt they were more motivated to learn using this type of system than advanced learners. This is important to note, especially if we take into consideration that object oriented programming in C++ course is one of the most difficult in the college.

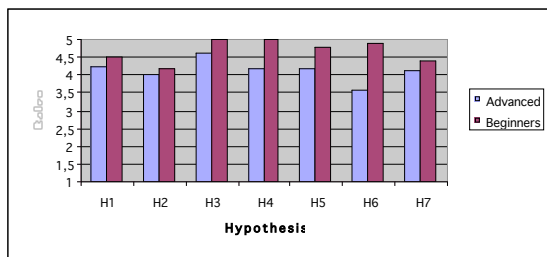


Fig.3. Assessment Framework Evaluation

The results support our research goals in that learners received a personalized assessment which they appreciated. The verbal feedback was also revealing; although some students with significant prior experience criticized minor aspects of the system, all of them hoped that the tool would also be used for other courses. This is confirmed through the results of H5, which go along with our objective of providing a generic framework that can be used for other courses if a domain description is specified. Overall, learners expressed positive reactions to the personalized assessment system. Yet their feedback also shows that they are still looking for more personalized support and points out the unnecessary time they sometimes spend finding out what they are looking for. In addition, many learners appreciated the fact that a personalized interface is provided, showing the recommended Pre- and Post-tests for the current learning resource based on their performance history, as confirmed by H6 results. They also expressed interest in getting a summary of the recorded performances, the learned resources and the solved assessment resources with their answers. The results of H7 show that learners do not only perceive the assessment framework as a useful tool in preparing for their exams, but have experienced it as a support tool in their education, by providing a personalized navigation through the learning resources. As mentioned earlier, a first version of the system was evaluated during this experiment. This version offered only multiple choice questions. Learners expressed their willingness to engage in tests that involve additional types of questions and might provide a better estimation of their knowledge of the course.

4. Conclusion and future Work

The results that emerged from the first experiment with the Personalized Assessment Framework are encouraging. In particular, they indicate that the framework does contribute positively to student learning and assessment of academic material in better terms than methods traditionally used in distributed learning environments. The first experiment with this tool has also uncovered a number of areas of

improvement. Future implementation and testing of the tool will be enhanced by incorporating new types of questions that include the testing of learner knowledge through the delivery of programming instructions. By including this new type of assessment resources, the framework will provide a computer-empowered evaluation system that significantly supports the personalization of assessment and learning in open environments, particularly for programming language courses. Furthermore, the ability to test this personalized assessment system with a larger number of learners will certainly pave the way for additional enhancement of the functionalities offered to learners

5. References

- [1] T. Berners-Lee, J. Hendler, and O. Lassila,, *The semantic web*, Scientific American, May 2001.
- [2] P. Brusilovsky and W. Nejdl, "Adaptive hypermedia and adaptive web". *Practical Handbook of Internet Computing*, M. Singh, Ed. CRC Press, 2004.
- [3] P. Dolog, N. Henze, W. Nejdl and M. Sintek, "Personalization in distributed elearning environments", in Proc. of the 13th International Conference WWW2004, New York USA, May 2004.
- [4] Dublin Core Metadata Initiative : <http://dublincore.org/>.
- [5] E.Gouli, K. Papanikolaou, and M. Grigoriadou "Personalizing Assessment in Adaptive Educational Hypermedia Systems" ", in Proc. of the 2nd International Conference of Adaptive Hypermedia and adaptive web-Based Systems, Malaga Spain, May 2002.
- [6] E. Guzman, and R. Conejo, "A Brief Introduction to the New Architecture of SIETTE", In Proc. of the 3rd International Conference on Adaptive Hypermedia, Eindhoven Netherlands, August 2004.
- [7] IMS Question and Test Interoperability Specification, V1.2: <http://www.imsglobal.org/question/index.cfm>.
- [8] L.Cheniti-Belcadhi, N.Henze and R.Braham "An Assessment Framework for eLearning in the Semantic Web", In Proc. of the 12th GI-Workshop on Adaptation and User Modeling in interactive Systems (ABIS04), Berlin Germany, October 2004.
- [9] LTSC Learning Technology Standards Committee. Learning Object Metadata: <http://ltsc.ieee.org>.
- [10] Personal Reader Framework: www.personal-reader.de
- [11] Question Mark Web: <http://www.questionmark.com>.
- [12] M. Sintek and S. Decker "TRIPLE- An rdf query, inference, and transformation language for the semantic web", In Proc. of 1st International Semantic web Conference, Sardinia, Italy, June 2002.
- [13] W3C. Semantic Web Activity: Resource Description Framework (RDF): <http://www.w3.org/RDF/2001>.

Personalized E-Learning in the Semantic Web

Nicola Henze

IVS – Semantic Web Group, University of Hannover, Germany
henze@13s.de

Abstract—This paper describes our idea for realizing personalized e-Learning in the Semantic Web. We have developed a framework for designing, implementing and maintaining Personal Learning Object Readers, which enable the learners to study Learning Objects in an embedding, personalized context. We describe the architecture of our Personal Reader framework, and discuss the possible authoring processes for creating Personal Learning Object Readers.

Index Terms—Personalization Services, Personalized e-Learning, Semantic Web.

I. INTRODUCTION

Personalization for the Semantic Web is still in its infancy. We are lacking flexible and re-usable personalization strategies which can be applied in various but similar contexts. One approach to overcome this problem is working towards personalization plug-ins: Services, which offer a certain personalization strategy, e. g. creating a guided tour, or recommending information, or annotating materials, etc. [11].

Within the Personal Reader project, we have developed a framework for designing, implementing, and maintaining *personalized* Web Content Readers. The Personal Reader framework makes use of recent Semantic Web technologies for realizing a Service-based environment for implementing and accessing Personalization Services. Several, distributed Services - for providing the user interface, for mediating between user requests and available Personalization Services, for user modeling, for providing personal recommendations and context information, et cetera, form the core of the Personal Reader framework. Prototypes of Personal Readers for e-Learning have been realized for the topics “programming in Java”, “Kobun”, and “Semantic Web” [11]. As a proof-of-concept of the underlying architecture of the Personal Reader framework, a Personal Reader for browsing scientific publications has been realized, too [1].

This paper is organized as follows: In the next section, we will outline the architecture of the Personal Reader framework. Afterwards, we describe the authoring process for creating Personal Readers for e-Learning, and discuss it by example of a Personal Reader for learning the Java programming language. A comparison with related work and a summary will conclude the paper.

II. ARCHITECTURE OF THE PERSONAL READERS FRAMEWORK

The question on how to enable personalization functionality in the Semantic Web can be regarded from different viewpoints, involving different disciplines, e. g. data mining, machine learning, Web graph analysis, collaborative approaches, and adaptive hypermedia. In our approach, we concentrate on methods and techniques

developed in the area of adaptive hypermedia (For an overview on methods and techniques of adaptive hypermedia, we refer the reader to [4]). An analysis and comparison framework, based on a logical description of adaptive *educational* hypermedia systems, has been presented in [13]. Here, some typical methods used in adaptive hypermedia systems, have been described as rules in first order logic. Required data (metadata about the documents, the users, as well as run-time data like observations about user interactions, etc.) have been identified and described.

The architectural outline for implementing the Personal Reader is a rigorous approach for applying Semantic Web technologies. A modular framework of components / Services - for visualizing the Personal Reader and providing the user interface, for mediating between user requests and available Personalization Services, for user modeling, for providing personal recommendations and context information, et cetera, is the basis for the Personal Reader.

The communications between all components / Services is syntactically based on RDF descriptions. E. g. the request for getting personal recommendations for a learning resource for a certain user is provided by an RDF description which is exchanged between the components mediator and personal recommendations. Thus each component is a Service, which is independent from the others and which can interact with them by “understanding” the RDF notifications they send (see Figure 1). The common “understanding” is realized by referring to semantics in the ontologies [12] used in the RDF descriptions which provide the valid vocabulary.

III. CREATING PERSONAL READERS FOR E-LEARNING

The Personal Reader framework offers off-the-shelf *Personalization Services* for e-Learning. These Personalization Services realize some of the adaptation techniques from the area of adaptive educational hypermedia identified and characterized in [13].

Authoring is a very critical issue for successfully realizing adaptive educational hypermedia systems. Thus, our aim is to have an easy authoring process which will produce learning materials which can be viewed with (some of) the off-the-shelf Personalization Services of the Personal Reader framework. As a guideline for our work, we established the following rule:

Learning Objects, course description, domain ontologies, and user profiles must be annotated according to existing standards (for details please refer to [11]). The flexibility must come from the Personalization Services which must be able to reason about these standard-annotated Learning Objects, course descriptions, etc.

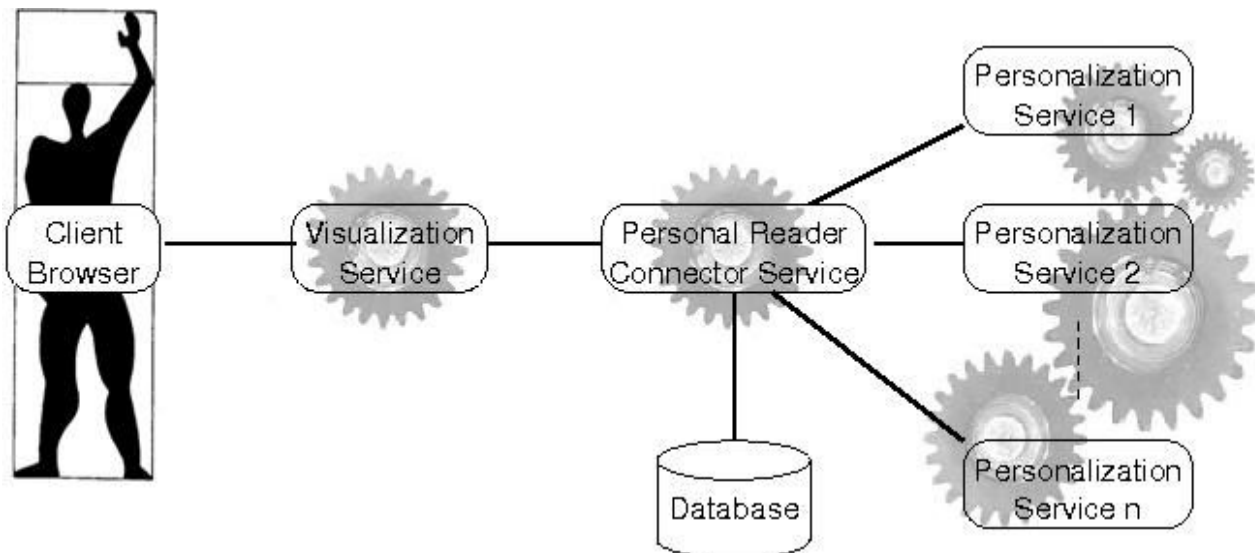


Figure 1: Architecture of the Personal Reader framework, showing the different components of the Personal Reader: Visualization (user interface), the Personal Reader backbone (consisting of the Connector Services, the Reasoning Service(s)), and some data-provision Services, for RDF data and for the connection with some database for storing user profile information.

For example, to create a new Personal Reader for an e-Learning course, the author of the course has to provide a metadata description of the new course in the language of RDF (Resource Description Framework [16]. Learning Objects in the course are annotated according to recent standards for Learning Objects like LOM [15] plus references to the basic Dublin Core standard [9]. The following code gives an example of such a course description for a course on Java Programming. The complete description is available at http://www.personal-reader.de/rdf/sun_java_tutorial.rdf.

```
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms#"
  xmlns:lom="http://ltsc.ieee.org/2002/09/lom-base#"
  xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#">
  <rdf:Description rdf:about=
    "http://java.sun.com/.../tutorial/index.html">
    <rdf:type rdf:resource="http://ltsc.ieee.org/2002/09/lom-
      educational#Book"/>
    <dc:title>The Java Tutorial (SUN)</dc:title>
    <dc:creator>
      <lom:entity>
        <vCard:FN>M. Campione</vCard:FN>
      </lom:entity>
      <lom:entity>
        <vCard:FN>K. Wallrath</vCard:FN>
      </lom:entity>
    </dc:creator>
    <dcterms:hasPart>
      <rdf:Seq>
        <rdf:li rdf:resource=".../tutorial/java/index.html"/>
      </rdf:Seq>
    </dcterms:hasPart>
  </rdf:Description>
  ...
  <rdf:Description rdf:about=
    "http://.../tutorial/java/concepts/message.html">
    <rdf:type rdf:resource="http://ltsc.ieee.org/2002/09/lom-
```

```
      educational#LO"/>
    <dc:title>What Is a Message?</dc:title>
    <dc:subject rdf:resource="http://www.personal-
      reader.de/rdf/java_ontology.rdf#OO_Methods"/>
    <dcterms:isPartOf rdf:resource=
      "http://.../tutorial/java/concepts/index.html"/>
  </rdf:Description>
  ...
</rdf:RDF>
```

The administration component (see Figure 2) of the Personal Reader framework provides an author interface for easily creating new instances of course-Readers: Course materials which are annotated according to LOM (or some subset of it), and which might in addition refer to some domain ontology (this is only optional), can immediately be used to create a new Personal Reader instance which offers all the personalization functionality which is - at runtime - available in the Personalization Services.

Figure 3 shows a screenshot of the Personal Reader for the Java programming course. This Reader displays the learning resources of the Sun Java Tutorial [7], a freely available online Tutorial on Java programming. It helps the learner to view the learning resources in a context: In this context, more *details* related to the topics of the learning resource, the *general topics* the learner is currently studying, *examples, summaries, quizzes*, etc., are generated and enriched with personal recommendations according to the learner's current learning state, as shown in figure 3.

A. Inside a Personalization Service

An example of a Personalization Service is a “detail viewer” which delivers details for some given Learning Object. This Service hosts – among some other rules – the following rule: Let *LO* denote some given Learning Object, and *LO_DETAIL* denote the detailed Learning Objects we are looking for. *LO_DETAIL* fulfills our requirements, if it covers some learning concepts *C_DETAIL* which are details of those learning concepts covered in *LO*, or if *LO_Detail* is a subconcept of *LO* in

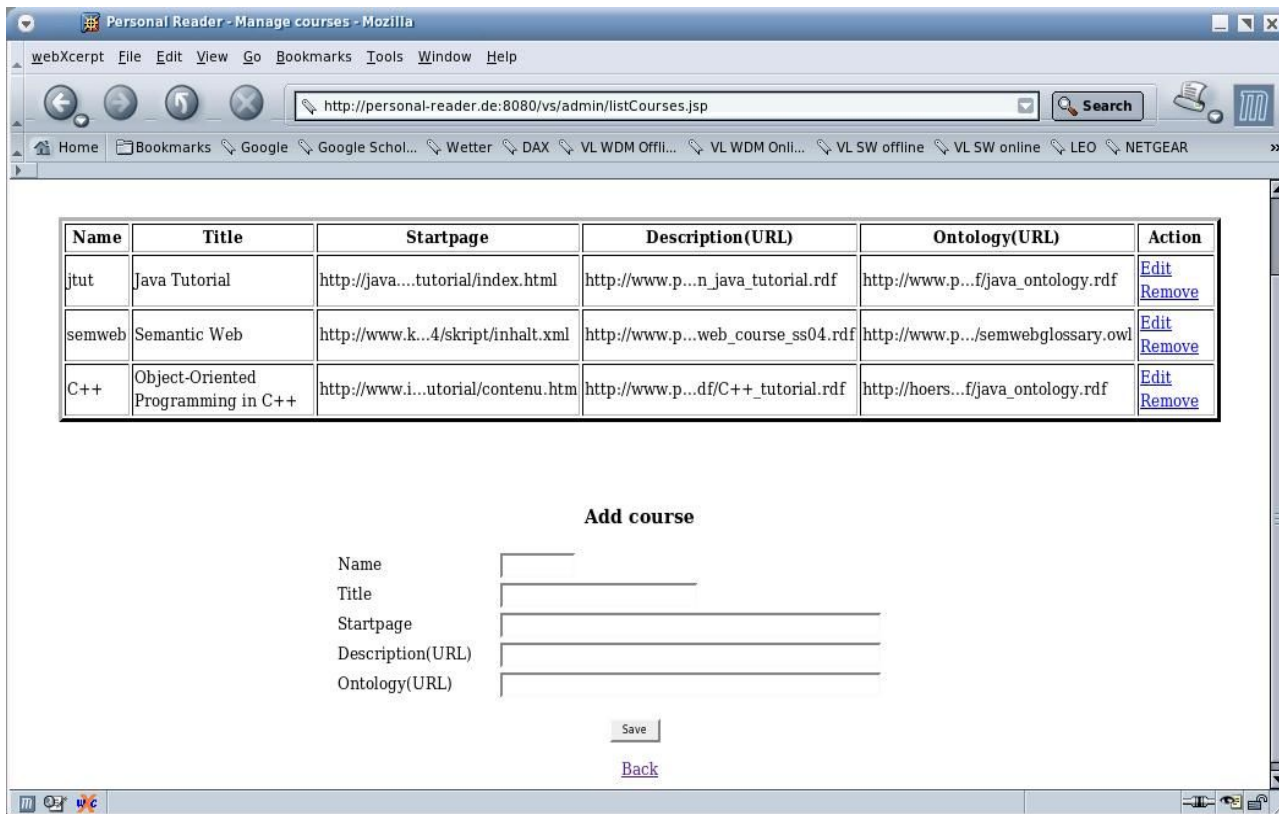


Figure 2: Screenshot of the administration interface of the Personal Reader, showing how to integrate new course materials.

the course structure. In the TRIPLE [17] language this rule looks as follows:

```
FORALL LO, LO_DETAIL
detail_learningobject(LO, LO_DETAIL) <-
  ( EXISTS C, C_DETAIL
    ( detail_concepts(C, C_DETAIL)
      AND concepts_of_LO(LO, C)
      AND concepts_of_LO(LO_DETAIL, C_DETAIL)))
  OR (upperlevel(LO_DETAIL,LO)).
```

B. Invocation of Personalization Services

Rules like the above are maintained within the Personalization Services. At runtime, each currently in the Personal Reader framework registered Personalization Service receives a request (in RDF) with information about the user, the page this user is currently visiting, etc., and generates e. g. personal recommendations for this user. These recommendations are coded in RDF, too, and passed back to the Personal Reader framework.

C. Advanced “Authoring”: Creating new Personalization Services

Another, advanced authoring possibility supported by the Personal Reader framework is to create new Personalization Services. As Personalization Services in the Personal Reader framework make use of standard Service technology, a new Service can simply register itself at a Web Service registry queried by the Personal Reader framework, and can then immediately receive requests from the mediator and answer requests.

RELATED WORK

Related work to our approach includes standard models of adaptive hypermedia like [2], recent personalization systems [8,10] as well as personalized learning portals [6].

Comparing our work with standard models for adaptive hypermedia systems like e.g AHAM [3], we observe that they use several models like conceptual, navigational, adaptational, teacher and learner models. Compared to our approach, these models either correspond to ontologies / taxonomies, to different schema describing teacher and learner profile, and to schema describing the navigational structure of a course. We express adaptation functionalities as encapsulated and reusable Triple rules, while the adaptation model in AHA uses a rule based language encoded into XML. AHA! provides the strategies for adaptation at the resources [2].

[10] focuses on content adaptation, or, more precisely, on personalizing the presentation of hypermedia content to the user. The technique used here is a slice-technique, inspired by the Relationship Management Methodology [14]. Both adaptability and adaptivity are realized via slices: Adaptability is provided by certain adaptability conditions in the slices, e.g., the ability of a device to display images. Adaptivity is based on the AHAM idea [3] of event-conditions for resources: A slice is desirable if its appearance condition evaluates to true.

[8] builds on separating learning resources from sequencing logic and additional models for adaptivity: Adaptivity blocks in the metadata of Learning Objects as well as in the narrative model, candidate groups and components define which kind of adaptivity can be realized on the current learning content. Driving force in these models are the candidate groups that define how to

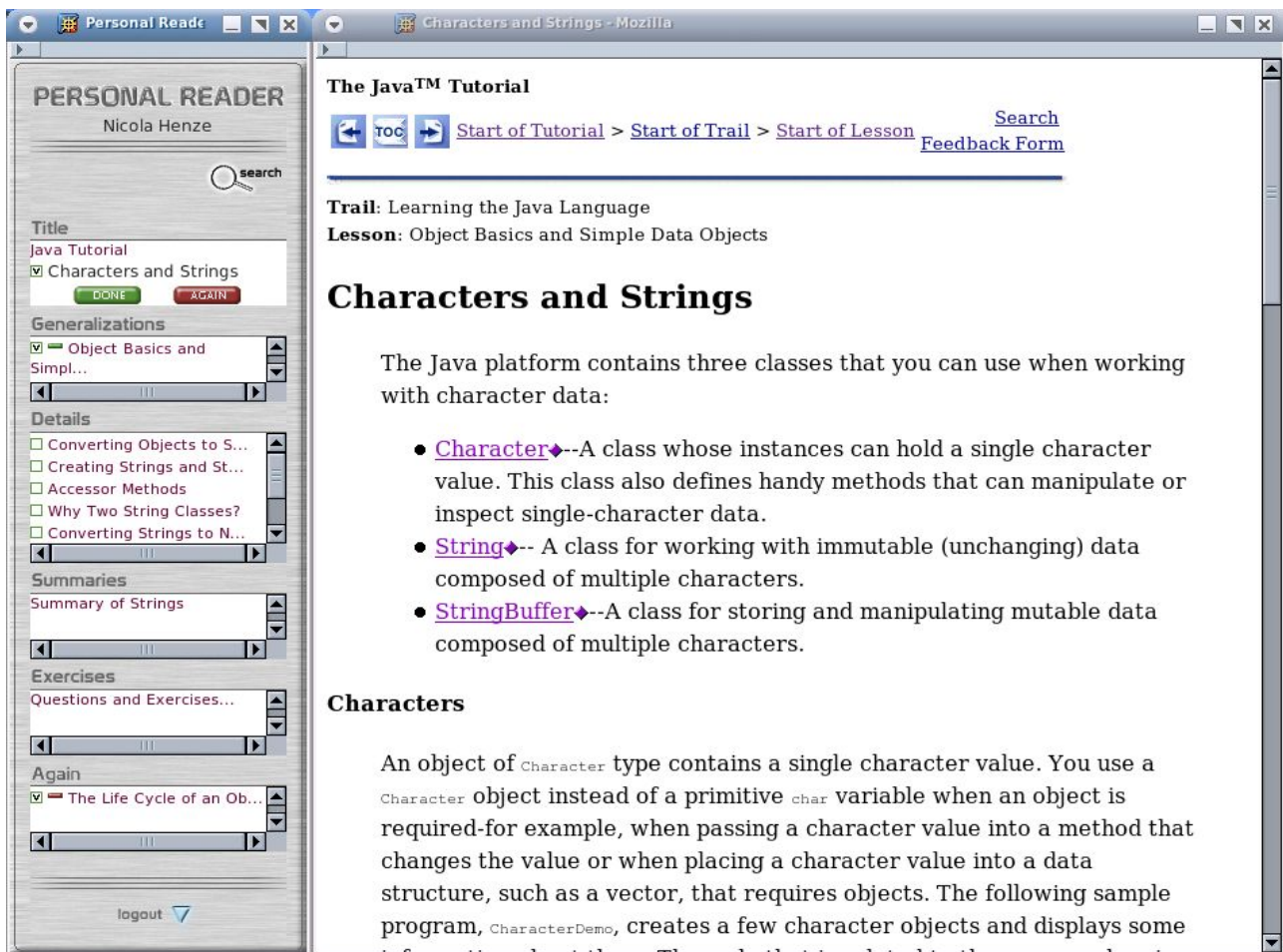


Figure 3: Screenshot of the Personal Reader for Java programming. The Personal Reader consists of a browser for learning resources (the reader part), and a side-bar or remote, which displays the results of the Personalization Services, e. g. individual recommendations for learning resources, contextual information, pointers to further learning resources, quizzes, examples, etc. (the personal part).

teach a certain learning concept. A rule engine selects the best candidates for each user in a given context. Adaptivity requirements are considered only in the adaptivity blocks.

Personalized learning portals are investigated in [6]. The learning portals provide views on learning activities which are provided by so-called {activity servers}. The activity servers store both learning content and the learning activities possible with this special content. A central student model server collects the data about student performance from each activity server the student is working on, as well as from every portal the student is registered to. In [5], also *value-added Services* are introduced in the architecture. The architecture in our approach is a simplification of the architecture presented here: We only consider *value-added Services*, and implemented our Personalization Services as these *value-added Services*.

IV. CONCLUSION

We have presented a framework for designing, implementing and maintaining adaptive {reader} applications for the Semantic Web. The Personal Reader framework is based on the idea of establishing personalization functionality as Services on the (Semantic) Web. The realization of personalization functionality is done on the logic layer of the Semantic

Web tower, making use of description and rule language recently developed in the context of the Semantic Web. We have tested the framework with an example reader, the Personal Reader for the Sun Java programming tutorial. Currently, we are using the framework to design a Reader for publications, and are investigating how learner assessment can be integrated to enhance the functionality for learning resources. The current state of the project can be followed at www.personal-reader.de.

ACKNOWLEDGMENT

This work has partially been supported by the European Network of Excellence REVERSE - Reasoning on the Web with Rules and Semantics (www.reverse.net).

REFERENCES

- [1] 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.

- [2] P. D. 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*, pages 240-246, Oct. 2002.
- [3] P. D. Bra, G.-J. Houben, and H. Wu. AHAM: A dexter-based reference model for adaptive hypermedia. In *ACM Conference on Hypertext and Hypermedia*, pages 147-156, Darmstadt, Germany, 1999.
- [4] P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87-129, 1996.
- [5] P. Brusilovsky. KnowledgeTree: A distributed architecture for adaptive e-learning. In *Proceedings of the Thirteenth International World Wide Web Conference*, New York, May 2004.
- [6] P. Brusilovsky and H. Nijhawan. A framework for adaptive e-learning based on distributed re-usable learning activities. In *In Proceedings of World Conference on E-Learning, E-Learn 2002*, Montreal, Canada, 2002.
- [7] M. Campione and K. Walrath. The java tutorial, 2003. <http://java.sun.com/docs/books/tutorial/>.
- [8] D. Dagger, V. Wade, and O. Conlan. Towards ``anytime, anywhere`` learning: The role and realization of dynamic terminal personalization in adaptive elearning. In *Ed-Media 2003, World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Hawaii, 2003.
- [9] Dublin Core, 2004. <http://dublincore.org/>.
- [10] F. Frasincar and G. Houben. Hypermedia presentation adaptation on the Semantic Web. In *Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)*, Malaga, Spain, 2002.
- [11] Nicola Henze. Personal Readers: Personalized Learning Object Readers for the Semantic Web. In *12th International Conference on Artificial Intelligence in Education, AIED'05*, Amsterdam, The Netherlands, July 2005.
- [12] N. Henze, P. Dolog, and W. Nejdl. Reasoning and ontologies for personalized e-learning. *ETS Journal Special Issue on Ontologies and Semantic Web for eLearning*, 2004.
- [13] N. Henze and W. Nejdl. A Logical Characterization of Adaptive Educational Hypermedia. *New Review of Hypermedia*. Vol. 10(1), 2004.
- [14] T. Isakowitz, E. A. Stohr, and P. Balasubrahmanian. RMM: A methodology for structured hypermedia design. *Commun. ACM*, 38(8), August 1995.
- [15] LOM: Draft Standard for Learning Object Metadata, 2002. <http://ltsc.ieee.org/wg12/index.html>.
- [16] Resource Description Framework (RDF) Schema Specification 1.0, 2002. <http://www.w3.org/TR/rdf-schema>.
- [17] M. Sintek and S. Decker. TRIPLE - an RDF Query, Inference, and Transformation Language. In I. Horrocks and J. Hendler, editors, *International Semantic Web Conference (ISWC)*, pages 364-378, Sardinia, Italy, 2002. LNCS 2342.

AUTHOR

Nicola Henze is Professor of Computer Science at the University of Hannover, Germany. She currently holds a Juniorprofessorship for Semantic Web at the Distributed Systems Institute (IVS). Her research interests are user modeling and adaptation, personalization in open hypermedia systems, reasoning on the Semantic Web, personalization in the Semantic Web, knowledge management and Web engineering.

Manuscript received May 17th 2006.

Configurable Web Services and the Personal Reader Agent

Personal Reader Team

July 5th 2006

The Personal Reader Framework enables us to develop and maintain Web Content Reader. In this architecture Web Services are primarily used as providers of RDF data, the content which is presented the end user of a Web Content Reader. With our new approach of Configurable Web Services we allow users to configure the data providing Web Services. Such configurations can be stored and reused at a later date. Thereby the Personal Reader Agent is the interface between Users and Configurable Web Services. The Agent allows selection, configuration and calling of the Web Services and further provides personalization functionalities like reuse of stored configurations which suit to the users interests.

Contents

1	Introduction	3
2	The Configuration Ontology	3
3	Configurable Web Services	6
3.1	MyEar Music Web Service	6
3.1.1	Configurable description of the MyEar Music Web Service	6
3.1.2	Internals of the MyEar Music Web Service	7
3.1.3	MyEar View	9
3.2	Personal Publication Reader Web Service	9
3.2.1	Core Functionality of the Personal Publication Reader Web Service	9
3.2.2	Configurable description of PPR Web Service	9
3.2.3	Special View for Results of PPR Web Service	10
4	Personal Reader Agent	10
4.1	Core Functionality	11
4.2	Personalization Functionality	14

1 Introduction

Within the Personal Reader project we already developed Web Content Reader like the *Personal Publication Reader*[2] which allows browsing publications in an embedded context. We also utilized and extended the SWAD-E Semantic Portal software[9] to provide a Personal Semantic Portal[5]. Whereas these approaches are fixed in terms of the type of data that is provided, we now introduce a more generic approach: Configurable Web Services and the Personal Reader Agent. The Personal Reader Agent is a Web Application which enables users to select, configure and call Configurable Web Services. These Semantic Web Services need a detailed description of how they can be configured and how they are accessible. According to this description the Personal Reader Agent generates an interface that allows to adjust the web services. The Agent further provides some personalization functionalities like reuse of stored configurations of web services which suit to the users interests. This article is structured as follows: In section 2 we introduce the *Configuration Ontology*, the ontological grounding of our approach. After introducing this ontology we can figure out Configurable Web Services considering a concrete Web Service as example (section 3). In section 4 we outline the architecture and implementation of the Personal Reader Agent. Finally we conclude our work and point out reasonable next steps (section 5).

2 The Configuration Ontology

The Configuration Ontology defines on the one hand the vocabulary that is needed to describe a Configurable Web Service and on the other hand the concepts that are required for the personalization functionalities.

Explanations for figure 1:

1. Core Configurable Vocabulary (needed to describe a Configurable Web Service):

Configurable Each Configurable Web Service defines one instance of this class. Therefor a **name** and a **description** has to be defined. Example:

```
(#MyEarConfigurable, name, "MyEar Configurable")
(#MyEarConfigurable, description, "Configurable things of my MyEar Music Web Service")
```

ConfigurableItem A Configurable consists of several ConfigurableItems. Example:

```
(#MyEarConfigurable, hasConfigurableItem, #DurationItem)
(#DurationItem, name, "Duration")
(#DurationItem, description, "Duration of a Song that should be taken into account by my Web Service.")
```

Input Every ConfigurableItem has at least one Input. We define two special Inputs: a **SelectionInput**, which allows only predefined values, and a **TextInput**, which allows arbitrary values. For an Input a **type**, a **minNumber** and a **maxNumberOfInputValues** has to be specified. Example:

```
(#DurationItem, input, #MinDurationInput)
(#MinDurationInput, description, "The minimum duration of a song (in minutes)")
(#MinDurationInput, type, http://www.w3.org/2001/XMLSchema#nonNegativeInteger)
```

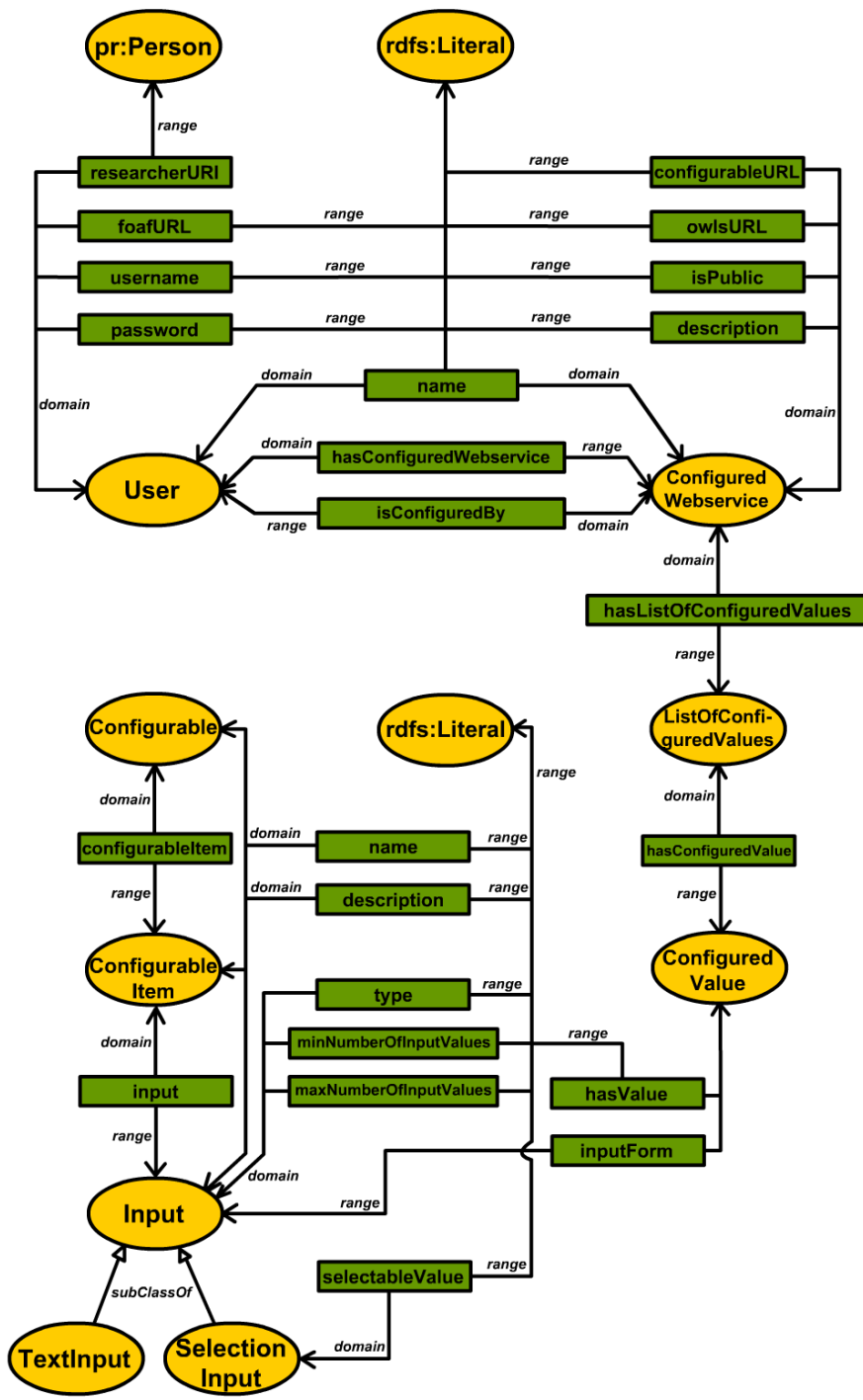



Figure 1: Configuration Ontology

```
(#MinDurationInput, minNumberOfInputValues, 0)
(#MinDurationInput, maxNumberOfInputValues, 1)

(#DurationItem, input, #MaxDurationInput)
...
```

2. User and their configured Web Services (needed to realize personalization functionalities):

User This concept models the users of our *Personal Reader Agent*. Therefor a **User** is featured with an **username**, **password**, **name** and a list of **ConfiguredWebservices** (**hasConfiguredWebservice**). To guarantee interconnection to other concepts that model a user we define the properties **researcherURI**, which points to the corresponding instance within our *Researcher Ontology*¹, and **foafURL**, which links the FOAF² description of the user. Example:

```
(#abelFabian, username, "fob")
(#abelFabian, password, "secret")
(#abelFabian, name, "Fabian Abel")
(#abelFabian, foafURL, "http://www.fabianabel.de/fabian.abel.rdf")
(#abelFabian, researcherURI, http://www.personal-reader.de/reverse#abelFabian)
(#abelFabian, hasConfiguredWebservice, #abelFabianMyEarJazzConfigWS)
(#abelFabian, hasConfiguredWebservice, #abelFabianPPRBioInformaticsConfigWS)
...
```

ConfiguredWebservice This concept is used to store configurations of Web Services made by a user. The properties **name** and **description** allow to describe the concrete configuration. The boolean property **isPublic** indicates whether a **ConfiguredWebservice** can be accessed and re-used by other users than the user who configured it (**isConfiguredBy**). **owlsURL** points to the OWL-S[6] description of the Web Service that was configured by the user and **configurableURL** points to the **Configurable** description. The values that belong to the concrete configuration are listed within the **ListOfConfiguredValues**. Example:

```
(#abelFabianMyEarJazzConfigWS, name, "Jazz Music")
(#abelFabianMyEarJazzConfigWS, description, "This configuration of the MyEar Music Web
Service effects the Web Service to aggregate
podcasting items that are related with Jazz.")

(#abelFabianMyEarJazzConfigWS, isPublic, "true")
(#abelFabianMyEarJazzConfigWS, isConfiguredBy, #abelFabian)
(#abelFabianMyEarJazzConfigWS, owlsURL, "...MyEar/rdf/MyEarOWLS.owl")
(#abelFabianMyEarJazzConfigWS, configurableURL, #MyEarConfigurable)
(#abelFabianMyEarJazzConfigWS, hasListOfConfiguredValues, #abelFabianMyEarJazzValueList)
```

ListOfConfiguredValues This is a list of the values that are configured by a user. Each **ConfiguredValue** has a **value** (range: typed Literals) and a reference to the **Input** (**inputForm**) which defines what is applicable in general. Example:

```
(#abelFabianMyEarJazzValueList, hasConfiguredValue, #abelFabianMyEarJazzValue1)
(#abelFabianMyEarJazzValue1, value, "3")
(#abelFabianMyEarJazzValue1, inputForm, #MinDurationInput)
(#abelFabianMyEarJazzValueList, hasConfiguredValue, #abelFabianMyEarJazzValue2)
...
```

¹The Researcher Ontology models the organizational structure of the REWERSE project: <http://www.personal-reader.de/rdf/ResearcherOntology.owl>

²Friend of a Friend: <http://www.foaf-project.org/>

3 Configurable Web Services

In the context of the *Personal Reader Agent* we use Configurable Web Services as RDF data provider. Such Web Services has to satisfy three requirements:

1. they need a Configurable description according to section 2
2. they have to be Semantic Web Services which means that they have a OWL-S description
3. they must be able to process RDF data (more precisely: `ListOfConfiguredValues`, section 2) in order to understand a Web Service call (see section 2 for details)

Below we illustrate concepts and implementation issues of Configurable Web Services by presenting the *MyEar Music Web Service* (section 3.1) and the *Personal Publication Reader Web Service* (section 3.2).

3.1 MyEar Music Web Service

The MyEar Music Web Service enables users to listen to their personalized *podcasting feed*. A podcasting feed is in fact a RSS 2.0 feed[10] whose items refer to audio files. Our MyEar Music Web Service searches the web for podcasting feeds that suits to the users interest and then combines items from different feeds to present a personalized podcasting feed to the user.

3.1.1 Configurable description of the MyEar Music Web Service

A part of the Configurable description is outlined in the examples of section 2. Overall there are four *ConfigurableItems*:

myEarKeywordItem A keyword that should be within an item of a podcasting feed. `myEarKeywordItem` is defined as followed:

```
(#myEarKeywordItem, input, #myEarKeyword)
(#myEarKeyword, rdf:type, #TextInput)
(#myEarKeyword, minOccurs, 1)
(#myEarKeyword, description, "Enter at least one keyword that should be within an item of
a podcasting feed, e.g. Jazz, Classic,..")
(#myEarKeyword, type, http://www.w3.org/2001/XMLSchema#string)
...
```

myEarItunesCategoryItem This item refers to a `SelectionInput` which only permits the selection of values that correspond to *itunes:category[1]*:

```
(#myEarItunesCategoryItem, input, #myEarCategory)
(#myEarCategory, rdf:type, #SelectionInput)
(#myEarCategory, selectableValue, "Music")
(#myEarCategory, selectableValue, "Public Radio")
(#myEarCategory, selectableValue, "Arts")
...
(#myEarCategory, minOccurs, 0)
(#myEarCategory, type, http://www.w3.org/2001/XMLSchema#string)
...
```

myEarDurationItem The duration item has two inputs which allow to specify minimum and maximum duration of audio files that should be included into the personalized podcasting feed (see also section 2):

```
(#myEarDurationItem, input, #myEarMinDuration)
(#myEarDurationItem, input, #myEarMaxDuration)
...
```

maxNumberOfGoogleCallsItem For the search of applicable podcasting feeds we use the *Google SOAP Search API*[4]. According to the Google API terms we only get 10 results per requests and thus the MyEar Music Web Service has to call the Google Search Web Service several times. By enabling the user to configure the maximum number of Google calls the user can affect the runtime of the MyEar process.

3.1.2 Internals of the MyEar Music Web Service


If the MyEar Music Web Service is called by the Personal Reader Agent then there is used only one parameter: RDF data which embodies the `ListOfConfiguredValues` (see last example of section 2). At first this RDF has to be parsed in order to extract the configured values. After that the following method is called:


```
public String getMusic(String keyword, String musicCategory,
                      Date dateFrom, Date dateTill,
                      int minDuration, int maxDuration,
                      int maxNumOfGoogleCalls){
//call the Gogle API maxNumOfGoogleCalls-times with keyword as query
...
//for each result of the Google API call:
processGoogleResult(result, resultChannel, musicCategory,
                    dateFrom, dateTill,minDuration, maxDuration);
...
}
```

Within the method `processGoogleResult()` the result list of the Google API call is filtered: Only each `rss:item` that fulfils the configured restrictions (e.g. `musicCategory`,...) is added to the `resultChannel` (`rss:channel`).

When the MyEar Music Web Service has finished the search a `resultChannel` is returned:

```
<rss version="2.0" ...>
  <channel>
    <title>Results of search...</title>
    <item>
      <title>Jazzatronic 2005</title>
      <link>http://www.bendingcorners.com/2006/jazzatronic_2005/</link>
      <description>
        BendingCorner's annual set of the past year's finest "jazzatronic"
        tunes. A collection of jazz-based electronically-influenced groovers
        with modern production techniques.
      </description>
      <pubDate>Sat, 29 Apr 2006 10:27:00 CEST</pubDate>
      <guid isPermaLink="false">http://www.bendingcorners.com/rss.xml</guid>
      <enclosure url="http://www.bendingcorners.com/2006/jazzatronic-2005-lo.mp3"
        length="33662849" type="audio/mpeg" />
    </item>
  </channel>
</rss>
```





Personal Music Aggregation


My Ear

[\[about\]](#) [\[browse\]](#)

Browse the results of the MyEar Webservice

Starfrosch - Jazz
Channel: <http://www.starfrosch.ch/pod/jazz.xml>

Pamela's Parade - Les ventilateuses [insub02]



Pamela-s Parade wurde 2000 in Genf als Drum und Sax Duo gegründet, später kam eine Posaune und ein Doublebass dazu.

Pamela-s Parade ist inzwischen ein Jazz Quartet beeinflusst von Folk und Free Jazz oder pulsierendem Funk.

<http://www.dincise.net/insubordinations/>

Download

Date: Wed Feb 22 12:30:15 CET 2006
URL: http://www.archive.org/download/INSUB02/insub02_Pamelas_Parade_01_les_ventilateu...
Duration: 35339074

In The Groove, Jazz and Beyond
Channel: http://www.lasternet.com/inthegroove/ITG_podcast.xml
A weekly Jazz radio show that airs on WHUS 91.7FM, Storrs CT. From the Jazz masters of past and present to emerging new artists performing jazz, fusion and funk. No smooth jazz here!


Great 88's
Here is some great piano jazz from both well known masters as well as little known talented jazz pianists.

Date: Sun Apr 23 05:37:07 CEST 2006
URL: http://pod-serve.com/audiofile/filename/1120/itg_20060422.mp3
Duration: 41300000

Swing is in the Air for April 16, 2006

More new releases and reissues


Direct download: mp3 (62mb | 90min)



Date: Sun Apr 16 23:30:00 CEST 2006
URL: http://media.libsyn.com/media/radiojazz/swing_20060416.mp3
Duration: 64826230

ReadingCorner: jazz-n-groove

My Ear Player



Actual Playing:
name/1120/itg_20060422.mp3




Figure 2: MyEar View

```

    <title>A Tour of California Artists</title>
    ...
  </item>
  ...
</channel>
</rss>

```

Here each `rss:item` is extracted from a separate podcasting feed, e.g. the first item originates from a podcast which is located at: <http://www.bendingcorners.com/rss.xml>. Thus at the moment the `guid` element is estranged to link the podcast.

3.1.3 MyEar View

Results of the MyEar Web Service can be viewed in a standard RDF browser or in the *MyEar View* which is tailored for these results. A screenshot of this application is shown in figure 2. The *MyEar Player* on the right hand side directly allows users to listen to the audio files that are referenced by the items of the result channel.

3.2 Personal Publication Reader Web Service

3.2.1 Core Functionality of the Personal Publication Reader Web Service

The Personal Publication Reader (PPR) demonstrates how to provide personalized, syndicated views on distributed Web Data using Semantic Web technologies. The application comprises several steps: In first step, the information about different publications is extracted from distributed heterogeneous sources and enriched with machine-readable semantics. Next step would be reasoning step. In this step, rules reason about the created semantic descriptions and additional knowledge-bases like ontologies and user profile information. In last step which is actually user interface creation, the result of reasoning step in the shape of RDF is interpreted and translated into an appropriate, personalized user interface. In other words, with several simple mouse clicks, end user is able to browse all relevant information regarding an specific publication. Relevant information contains author's contact information, other publications of authors, different working groups, current publications in a specific working group and so on. For an online demo of PPR Web service, refer to [8].

3.2.2 Configurable description of PPR Web Service

As we have seen in previous sections, we need a Configuration-Description for a configurable Web service. Therefor we present a Configuration-Description for PPR. This Configuration-Description has been developed in two versions. The first version of Configuration-Description is based on the current functionalities of PPR. Generally, in current implementation of PPR, we send a RDF description including title of publication and PPR responses with all relevant information regarding this publication. The first version of PPR Configuration-Description has following main triples:

```
(#PPRConfigurable, hasConfigurableItem, #PublicationTitleItem)
```

```
(#PublicationTitleItem, input, #PublicationTitle)
(#PublicationTitle, description, "The title of the current publication")
(#PublicationTitle, type, http://www.w3.org/2001/XMLSchema#string)
(#PublicationTitle, minOccursOfInputValues, 1)
(#PublicationTitle, maxOccursOfInputValues, 1)
```

In the second version of Configuration-Description, we decided to use **Author name**, **Working group**, **Publication year**, and **Publication origin** as configurable items. In other words, we have following triples as configurable items:

```
(#PPRConfigurable2, hasConfigurableItem, #PublicationAuthorNameItem)
(#PPRConfigurable2, hasConfigurableItem, #PublicationWGItem)
(#PPRConfigurable2, hasConfigurableItem, #PublicationYearItem)
(#PPRConfigurable2, hasConfigurableItem, #PublicationOriginItem)
```

Each configurable item has its own properties. In a human language we can say: **#PublicationAuthorNameItem** has an input **AuthorName** and should occur exactly once in a request. **#PublicationWGItem** is a list of eight active working groups in REW-ERSE project. A request can contain at least zero and at most eight working groups. **#PublicationYearItem** has an input **Year** which indicates the publishing year. Finally, **#PublicationOriginItem** is a list of current twenty cities which publication information is gathered from their academic institutes.

3.2.3 Special View for Results of PPR Web Service

In an architectural overview, PPR is composed of three kinds of Web services: **Personalization services**, **Connector services**, and **Visualization services**. Visualization services are responsible for presenting the result of query to end use according to output device type and user settings. Beside RDF which is actually the raw format of output, the other supported format is currently HTML and it can be shown using any kind of Web browser. For an online demo of PPR application, refer to [8].

4 Personal Reader Agent

The Personal Reader Agent is on the one hand a kind of wizard that allows to select, configure and call the Configurable Web Services (*Core Functionality*) and on the other hand it provides the management of users and their saved configurations (*Personalization Functionality*).

For technical purpose the Agent is a J2EE Web Application that adheres to the *Model View Controller* approach. Figure 3 gives an overview of the Agent's architecture (or rather the package structure). General description of the Agent's packages:

reverse.agent The **AgentManger** is implemented as a singleton and is a specific extension of the session management. It memorizes objects that are relevant for the actual sessions like *selected Web Services of a user identified via SessionID*.

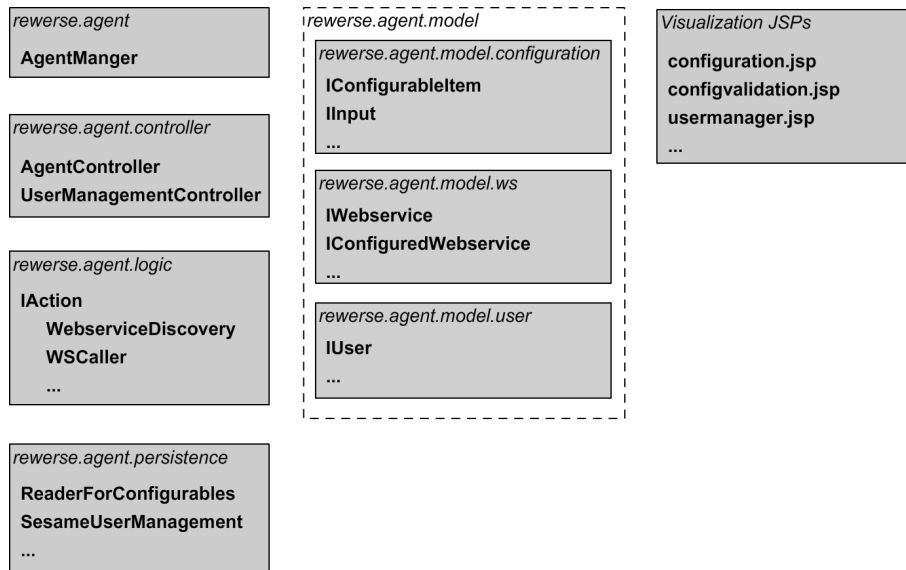


Figure 3: Architecture - Overview of Agent packages

rewerse.agent.controller This package contains the main controller (*HttpServlets*) of the Agent application: **AgentController** (responsible for core functionality) and **UserManagementController** (responsible for personalization functionality).

rewerse.agent.logic All actions like **WSCaller**, **WSConfigurationValidator** etc. implement the Interface **IAction**.

rewerse.agent.persistence The persistence package contains classes that..

- ... read in the OWL-S description and Configurable description of the Configurable Web Services and]
- ... provide access to a Sesame Repository[3] which stores user details and configurations made by users.

rewerse.agent.model Methods of the persistence layer return generally instances of the model package. We deal with three different kind of objects: Users, Web Services and Configurable descriptions. Thus we provide three packages containing Java models for these different types of objects.

Visualization JSPs Our presentation layer consists of a set of *Java Server Pages* (JSPs) which visualize the model objects that are the actual point of interest.

4.1 Core Functionality

The Controller that is responsible for the core functionality is the **AgentController**. It affords the following steps:

1. **Discovery** In this step the `AgentController` calls the `WebserviceDiscovery` action which requests the OWL-S descriptions of the Configurable Web Services that are registered at our simple UDDI. Finally the OWL-S descriptions are prepared for a selection by the users, according to figure 4.
2. **Configuration** After the first step the Agent reads in the Configurable descriptions of the selected Web Services (`ReaderForConfigurables`). Out of a corresponding Java object (`reverse.agent.model.configuration.IConfigurable`) a JSP generates HTML Forms that are illustrated in figure 5.
3. **Web Service Call** After all selected Web Services are configured without violating the restrictions defined in the corresponding Configurable descriptions (e.g. `maxNumberOfInputs`, `type`, ...), the Agent (`WSCaller`) is ready to call the Web Services (see figure 6). To ease the Web Service call we use the *OWL-S API*[7] provided by *mindswap*.
4. **Presenting the results** This step is not part of the Agent application but rather a task that can be done by a common RDF browser like *Piggy Bank*³ or an application that provides a special view for a certain RDF data (e.g. *MyEar View* visualizes *RSS 2.0 feeds*).

Step 1: Selecting a Webservice

Please select at least one of the following Webservices:

PPR
The PPR Webservice generates a personalized Research Information.

MyEar
The MyEar Webservice generates a personalized Podcasting feed.

Progress

1. Selection of Webservices 2. Configuration of Webservices 3. Ready to call Webservices 4. Browse Results

Figure 4: Step 1 - Selection of Configurable Web Services

³Piggy Bank - Firefox extension that allows browsing RDF data - <http://simile.mit.edu/piggy-bank/>

Step 2: Configuration of Webservice

Configure the selected Webservice to your needs...

MyEar Configuration

Configurable Things of MyEar Service

Query Keywords

Type in your keywords (seperated with blanks) that should be within the podcasting item, e.g.: Jazz Swing

Value:

Enter at least 1 values.

itunes category

An itunes:category... [Info: This restriction may lead to few considered items or in extreme case also to timeouts!]

Duration

Specify a range for the length of the audio files. If you are searching for radio-like shows then you should enter a high Minimum Duration. Otherwise if you are searching for single songs then we recommend you to enter a low Maximum Duration.

Info: Some podcasting producer do not specify the duration of their audio files. If you do not want to pass over these podcasts then leave both fields blank.

Maximum Duration
The maximum duration of audio files (in minutes).

Minimum Duration

Figure 5: Step 2 - Configuration of Web Services

Step 2: Configuration of Webservice

MyEar Configuration - Values you entered/selected

Query Keywords	Jazz
itunes category	Music
Duration	Maximum Duration: 60 Minimum Duration: 15
Maximum Number of Google API Calls	2

If you click the *confirm*-Button the Webservices you selected and configured are executed. Otherwise click the *back*-Button to edit your inputs.

Progress

1. Selection of Webservices 2. Configuration of Webservices 3. Ready to call Webservices 4. Show Results

Figure 6: Step 3 - Ready to call Web Services

• Click on the "confirm + save"-Button to confirm and save your entries. Saving allows you to re-use your configuration at a later date.

MyEar Configuration - Values you entered/selected

Duration	Maximum Duration: 20 Minimum Duration: -
Maximum Number of Google API Calls	2
itunes category	Music
Query Keywords	Jazz

If you click the confirm-Button the Webservices you selected and configured are executed. Otherwise click the back-Button to edit your inputs.

Save your configuration of the Webservice

Saved configurations can be used at a later date to call the Webservice in a personalized way. It is also possible to adjust your configuration later. You have to enter at least a significant name for your configuration. Further you can enter a short description and select whether other users can utilize your configuration or not (standard is not).

Name: (e.g. MyEar - Jazz)

Description: (e.g. With this configuration the MyEar-Webservice returns a Jazz-Podcasting Feed...)

publish?: (If checked then other users can utilize your configuration)

Progress

1. Selection of Webservices 2. Configuration of Webservices 3. Ready to call Webservices 4. Entered Results

Figure 7: Saving Configurations - Entry of meta description about a configured Web Service

4.2 Personalization Functionality

Registered users have some options which lead to a more personalized Agent, they are enabled to...

...save configurations: As in section 2 outlined the ontological model behind *saved configurations* is *ConfiguredWebservice*. At this users can specify name, description and *isPublic* on their own, *owlsURL* and *configurableURL* is set by the Agent and the *ListOfConfiguredValues* arises from the configuration step which is also performed by the users. The HTML input form that allows entry of such meta information about configured Web Services is presented in figure 7.

...re-use their own configurations: In order to allow users a faster access to the Configurable Web Services they can call these services also with a saved configuration as illustrated in figure 8. Further the Agent provides some management functionality for configured Web Services (*view*, *edit* and *delete*).

...re-use recommended configurations of other users: If a *ConfiguredWebservice* is marked as *isPublic* then it can be re-used also by other users than the author. Therefor the Agent allows the listing of configurations that might be relevant for a user. To determine relevant configurations the Agent utilizes relations between users that are defined inside the *Researcher Ontology* or *FOAF* description. The Agent proceeds as follows:

Your Configured Web Services

List of Web Services you have configured using the Personal Reader Agent is shown below. You have the opportunity to view, edit or delete your Configured Web Services. Further you can re-use a Configured Web Service by clicking on call.

Name	Description	Is Public?	Actions
Die Testkonfiguration	Dies ist eine Testkonfiguration, die nichts bewirkt...	false	view edit delete call
MyEar - Jazz	A simple search for podcasting feeds related with Jazz...	true	view edit delete call
Rock - MyEar	A bissel Rock...	true	view edit delete call

If you have problems then mail us: readerteam@kbs.uni-hannover.de

Figure 8: List of configured Web Services

Relation within Researcher Ontology: The Researcher Ontology defines persons and their involvements in working groups. If two persons (users) are involved in the same working group then the Agent suggests that configurations made by *User A* are also interesting for *User B*.

Relation within FOAF description: FOAF defines among other things the relation (*Person, knows Person*). This relation can be used by the Agent to list configurations of other persons the user knows.

5 Conclusion and Outlook

With the Personal Reader Agent and Configurable Web Services we provide a dynamic approach to aggregate RDF data. With the Agent's personalization functionality we also allow to personalize the access to the Configurable Web Services.

Based on the current state of implementation there are still some open issues and conceivable extensions... [to be continued]

References

- [1] Apple Computer, Inc.
Podcasting and iTunes: Technical Specification, 2006
<http://www.apple.com/itunes/podcasts/techspecs.html>
- [2] R. Baumgartner, N. Henze and M. Herzog
The Personal Publication Reader: Illustrating Web Data Extraction, Personalization and Reasoning for the Semantic Web. European Semantic Web Conference ESWC 2005, Heraklion, Greece, 2005
<http://www.kbs.uni-hannover.de/Arbeiten/Publicationen/2005/eswcbhh.pdf>
- [3] J. Broekstra, A. Kampman and F. van Harmelen
Sesame: A Generic Architecture for Storing and Querying RDF, International Semantic Web Conference 2002, Sardinia, Italy, 2002
<http://openrdf.org/doc/papers/Sesame-ISWC2002.pdf>
- [4] Google
Google SOAP Search API, 2006
<http://www.google.com/apis/>
- [5] N. Henze and F. Abel
User Awareness and Personalization in Semantic Portals, International Semantic Web Conference, 2005
<http://reverse.net/publications/download/REWERSE-RP-2005-121.pdf>
- [6] D. Martin et. al.
OWL-S: Semantic Markup for Web Services, W3C Member Submission, 2004
<http://www.w3.org/Submission/OWL-S/>
- [7] mindswap, E. Sirin
OWL-S API, 2004
<http://www.mindswap.org/2004/owl-s/api/>
- [8] *Personal Reader Project*
<http://www.personal-reader.de/>
- [9] D. Reynolds, P. Shabajee, S. Cayzer and D. Steer
Semantic Portals Demonstrator - Lessons Learnt, SWAD-Europe deliverable 12.1.7, 2005
http://www.w3.org/2001/sw/Europe/reports/demo_2_report/
- [10] RSS Advisory Board
Really Simple Syndication 2.0 Specification, UserLand, 2002
<http://www.rssboard.org/rss-2-0/>

UMService: An application independent User Modelling Service

Ingo Brunkhorst², Nicola Henze¹, Daniel Krause^{1,2}

¹ Distributed Systems Institute
Knowledge Based Systems
University of Hannover
D-30167 Hannover, Germany
henze, krause@kbs.uni-hannover.de
² L3S Research Center
University of Hannover
D-30539 Hannover, Germany
brunkhorst@l3s.de

Abstract. In distributed systems, two main strategies for user modelling are widely used: A per-application based user modelling approach and a centralized user modelling server. We propose an approach, the UMService, which combines the advantages from both strategies: Being on the one hand a domain-aware centralized user modelling service that, on the other hand, enables inter-application user model sharing. In the second part, we compare our approach with different state-of-the-art approaches. Therefore, we define important criteria for user modeling systems in distributed systems.

1 Introduction

In the last years, systems were designed to allow creation of semantically rich web applications, based on technologies from the semantic web domain, like RDF³, and on a service oriented architecture approach, by using semantically enriched Web-Services. Examples of these systems are the Internet Reasoning System (IRS-III)⁴ and the Personal Reader Framework (PR)⁵, which both allow the creation of rich web applications based on semantic web services. However, the primary focus of these systems is to facilitate automatic selection, composition and even choreography of semantic web services. As they are high level frameworks, they do not yet support all of the special needs necessary for other aspects of building *personalized* and *adaptive* semantic web applications.

In this paper we describe an additional architectural component for our Personal Reader Framework, with the single purpose of allowing domain-independent and application-independent management of a user model. Basically, the PR consists of a set of heterogeneous components: A centralized system for locating and composing services according to the users requirements, and as the basic building blocks for personalizing

³ <http://www.w3c.org/rdf>

⁴ <http://kmi.open.ac.uk/projects/irs/>

⁵ <http://www.personal-reader.de>

and adapting web content, small independent services —Personalization Services— that each provide personalization functionality. Personalization Services are suitable for a single domain or are domain-independent and allow the building of complex applications by combining them with Syndication Services for creating user interfaces for displaying web content.

All of the applications realizable within these frameworks need to have access to a user model, storing generic user profile information, user preferences, and even session-related records. Furthermore, the information must be easily accessible by the services, but still protected against forms of exploitation by untrusted services or simply by respecting the users' stated privacy concerns.

The suggested *User Modelling Service* allows services to store and update user-dependent information relevant for adapting and personalizing the access and presentation of content in each step of accessing services, also allowing persistent storage of user preferences to enable services to be more adaptive. For example, in the domain of educational systems, the semantic web technologies are widely used for describing not only learning resources itself, but also processes and workflows of learning activities. Within the Personal Reader Framework, personalization services for presenting and working with e-learning material are already implemented, as well as a service for planning of personalized curricula [1].

Using a centralized and common user model allows services to adapt better to the users progress within the whole system. By taking into account the preferences and interests of the user, which were discovered by the presentation service in form of “most favorite topics”, this can lead to the creation of a better suited personal curriculum plan for the student. Although other ways of managing the user's context across the different personalization services are possible, a common centralized user modelling service allows the easy aggregation and querying of relevant application-dependent context information in a single place.

In section 2 we will list some of related work in the domain of user modelling, followed in section 3 by a description of our Personal Reader Architecture. Section 4 gives a detailed description of the proposed User Modelling Service, followed by a list of possible Scenarios (Section 5). Conclusions and Future Work is given in section 6.

2 Related Work

Every application that wants to adapt it's behavior to a user, requires to model the users preferences, interests and needs, which is commonly referred to as the *User Model* [2]. User models can be application specific, as in almost every currently deployed rich web applications, e.g. Amazon, Google. These model usually are specifically tailored for the single purpose of adapting the application in their specific domain. A different approach is given with the *General User Model Ontology* [3], which was influenced by other approaches like *UserML* [4].

As the GUMO model is implemented as an OWL Ontology, it is well suited for use in the RDF-based Personal Reader Framework specific implementation, and since our approach is also geared towards general user modelling, we encourage the use of GUMO for modelling the domain-independent parts in our user model. In fact, the

*user model server*⁶ is similar to our approach, as it provides a web service to manage a centralized user model, however it does not allow the same fine grained control of access protection. Additionally, more complex requests are needed to update, manage and query the user model, or parts of it.

Traditionally, adaptive applications each manages its own user model, closed and disconnected from other applications. However, there exist mediation approaches [5], where an application can enrich it's own model by importing, translating and aggregating partial User Models from other —possible related— services. Our approach allows an application to enrich the general user model or add application specific user information to the model, in a way that the information is available to other services.

3 The Personal Reader Architecture

Our approach for a Semantic Web browsing application offers the user a uniform entry point to access the Semantic Web, and in particular to Web services in the Semantic Web. It has been realized as part of the *Personal Reader Framework* [6–8] which offers an environment for designing, implementing and realizing Web content readers in a service-oriented manner (see Figure 1):

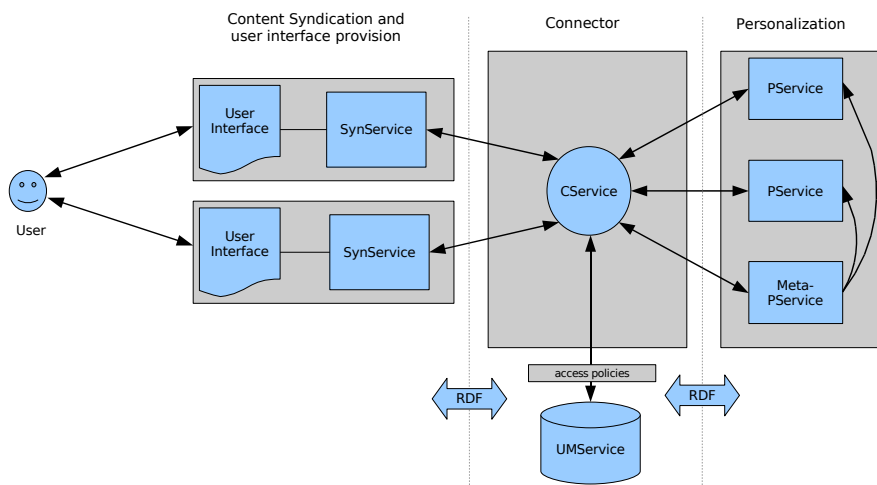


Fig. 1. Overview of Personal Reader architecture

- Web services deliver personalized recommendations for content, extracted and obtained from the Semantic Web. Using Semantic Web techniques they describe their offered functionality in a machine processable format. Optionally, they can return

⁶ <http://u2m.org>

visualization templates that can be used to create user interface snippets for presenting the results of the Web services. We call these kind of Web services *Personalization Services*, *PServices* for short.

- *Meta PServices* can be employed to have single entry points to cooperating or concurrent PServices. For example, a music recommender Meta Personalization Service orchestrates different PServices delivering personalized music recommendations. In an internal processing step the music Meta-Web service filters and orders all the resulting recommendations of the different PServices into one single result.
- For syndicating the results of PServices and for creating appropriate user interfaces, *SyndicationServices*, or *SynServices* for short, are responsible for displaying the results of several PServices and/or Meta PServices to the user. Each SynService provides such a user interface endpoint to a certain domain or task, and allows the user to benefit from many PServices simultaneously, which are selected, combined and customized as the user wishes. SynServices can be realized as RDF browsers, can implement their own RDF processing interface, or they can make use of visualization templates provided by the different PServices and Meta PServices.
- For enabling the whole process, a core information provider – a user modelling service, *UMService* for short, deriving appropriate user profiles – is essential. In our architecture, the UMService realizes a centralized approach for user modelling, maintaining and protecting information about a user on behalf of this user.
- From a technical point of view, another component is required to maintain the communication between the SynServices providing the user interface, the PServices, and the UMService. This is the so-called *Connector Service* (*CService* for short) which harvests Web service brokers, collects information about detected PServices (for discovery, selection, customization, and invocation), and for organizing the communication between all involved parties, including requests to the UMService.

With the centralized CService component the Personal Reader Architecture allows Web Service authors to plug in their SynServices and (Meta-)PServices into the framework and hence create a distributed and independent network of P- and SynService. The plug-in step is easily done by registering the new service at a UDDI repository.

The benefit for Web Service authors is, that they can directly collaborate with all other Web Services without having to know them before. E.g. an author can create a SynService, describe which data sources the service requires and the Personal Reader Framework provides the PServices that fulfil the described requirements. Additionally, new Web Services can directly – after invocation by a user – access the UMService and therefor use the user profile to adopt their output according to a user's preferences.

4 The Personal Reader User Modelling Service

The UMService is a centralized component in the Personal Reader Architecture. It offers an interface for P- and SynServices to access, store and modify user profiles. Being centralized it enables different Services to share user profiles. Hence even services the user has never invoked before can personalize their output according to a user's profile, which lessens cold start problem. The second important issue is that the UMService

is not under the control of the applications, represented by the P- and SynServices but under control of the user. That means that the user can control which Web Service is allowed to access or modify which properties within his user profile. Additionally, this approach has the advantage that even competing service providers, that normally never share data, now have to share data with each other which results in a higher benefit for the user.

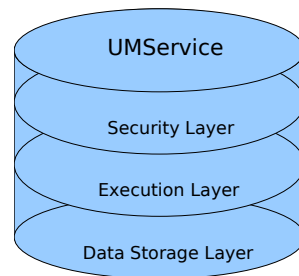


Fig. 2. Architecture of the User Modelling Service

The architecture of the UMService is divided into three components (see figure 2): The user profiles are stored in a RDF database that is accessed by the *Data Storage Layer* of the UMService. It is realized as an interface providing methods to query and update user profiles. In our current demo-application, the Agent⁷, we implemented the interface for querying a SESAME⁸ RDF data base. The *Execution Layer* processes requests from the P- and SynServices and generates the responses. Therefore, it queries the Data Storage Layer to receive the appropriate parts of the user profile. As the user profile data is stored as RDF data and the response are also RDF formatted messages, there is not much postprocessing of the query results necessary. In all web-related contexts, policies play an important role [9], and are also relevant for our UMService. Privacy policies are used to protect the information in the user model. The *Security Layer* checks if a service is allowed to access or modify the data denoted in its request. Only if the service is allowed to access the data, the Security Layer forwards the request to the Execution Layer.

4.1 Data storage

The user profiles are stored in an RDF database accessed by the Data Storage Layer. As storage format we use extended triples described in the General User Model Ontology (GUMO) [10, 3]. Furthermore, we use a modified subset of situational statements which offer a very expressive set of data and metadata that is required for storing user profile information.

⁷ <http://www.personal-reader.de/agent/>

⁸ <http://www.openrdf.org/>

The set of expressions in the GUMO's situational statements are reduced by the data we cannot generate, e.g. location information as we normally do not have appropriate sensors. Furthermore, we do not include privacy data as this is handled in a separate layer, namely in the Security Layer of the UMService, and logically does not belong directly to the user profile. We redefine the confidence value as a combination of two single confidence values: The first confidence value, `assumedConfidence`, can be freely chosen by the Web Service that generates the user profile entry. It represents the confidence of the Web Service regarding this statement. The second confidence value, `serviceConfidence`, is a per-Web Service value, that denotes the user's trust in the Web Service which made the statement.

4.2 Access Rules

In the UMService, the authentication and authorization process is handled by the Security Layer. If a request appears, the Security Layer checks if the sender is allowed to perform the task. Only if this check is successful (the Web Service is allowed to perform the task) the request is forwarded to the execution layer of the UMService. The check is performed with a simple lookup in a Web Service / property table. This property table stores the information if a Web Service *X* is allowed to access property *Y*

- never
- once
- always within the current session
- always

If the access right was set to *once* the request is performed and the entry is deleted in the Web Service / property table. If the access right is set to *always* or *always within the session* the request is performed without a modification of the Web Service / property table. When *never* is given as access right, the Security Layer sends an *access denied* response to the requesting Web Service. If no value is given in the Web Service / property table the UMService asks the user directly which access right to give to Web Service *X* regarding property *Y*.

Property Classes The UMService has to store properties from many different domains. This results in a huge amount of properties that have to be maintained in the UMService's Web Service / property table. E.g. if a Web Service tries to access 20 different properties the user is asked 20 times to define the access rights for the property.

To simplify the maintenance of access rights we use a high level Ontology that classifies the properties hierarchically. E.g. we define a class *confidential bank data* that contains properties like bank identification code, credit card number, and so on. Instead of forcing the user to allow the access for every single property, the user can allow the access to all properties that are instances of a class. Or, using the class hierarchy of the properties, the user can grant the access right for all instances of a class and for all instances of all the subclasses.

By using an Ontology defined in OWL/RDFS other Ontologies can easily classify their properties according to our Property Ontology by using `equivalenceClass`

expressions. Therefore, new properties can easily be added to the existing classification without requiring updates in our Ontology.

The defined property classes are added twice to the Web Service / property table to represent enabled and disabled recursive inclusion of subclasses.

Web Service Classes We allow all kinds of PServices and SynServices to register at the Personal Reader Framework. Hence, the UMService has to handle a large amount of Web Services and the problem is the same as with properties: If a user wants to invoke a SynService that invokes 19 PServices the user has to define 20 times which Web Service is allowed to access which properties. Therefore, we use the same approach as for properties: We classify our Web Service.

While we can find a classification that groups properties having the same degree and that is valid for all users this is not possible for Web Services: Every user has a different trust in different Web Services. Therefore, we use a user generated classification. The user can define different trust classes. These trust classes are arranged like a stack: Class i subsumes class $i - 1$ and is a subclass of class $i + 1$ and so on (see figure 3). Then, the user can define for every class which Web Services belong to which class. Furthermore, the user has to define which access rights to which properties and property classes the Web Service class has got. Because of the stack architecture a Web Service that is an instance of the class i receives all access rights granted to the classes $1..i$.

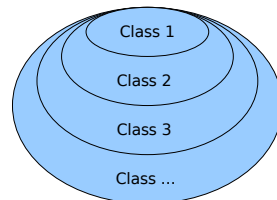


Fig. 3. The Web Service class hierarchy

This approach enables an easy configuration of the requested access rights: If a request arrives, that requests to access a set of properties we just need to search in which Web Service class the access right exists. Afterwards, we calculate which smallest class i includes all required rights and ask the user if he agrees to classify the Web Service into this Web Service class. Therefore, all access rights of one Web Service can be configured with one mouse click.

Automatic Classification There are several independent organizations, like the Better Business Bureau⁹ that rate different web sites. In the future, these organizations can be

⁹ <http://www.bbb.org/>

engaged to also rate and control Web Services. With the help of classification rules a user can define that Web Services should be classified automatically:

- good rated Web Services are automatically classified into a higher Web Service class
- poor rated Web Services are classified into the lowest class where they have no access to any user profile properties
- unknown Web Services receive a standard classification

Other collaborative rules can also be defined in this framework: If my friend allows the service to access his user profile I also automatically allow the Service to access my user profile.

This automatic classification can —if classification organizations' have an appropriate coverage of the invoked Web Services— reduce the amount of user interaction.

Security Object Both, SynServices and PServices need to access the user profile, stored by the UMService. On the one hand to retrieve user data for adapting the output to a user and on the other hand to store new or update existing information about the user after the user has interacted with the Syn- or PService.

UMService access by a SynService It is quite simple to identify the request to the UMService from an authorized SynService: The user knows the SynService as he accessed it directly. Hence, a simple and secured request (the *Security Object*) to the UMService sent by a CService is given by figure 4.

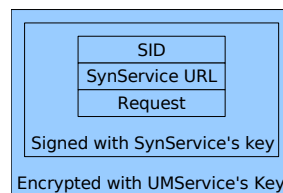


Fig. 4. The simple Security Object for SynServices

The SynService signs an object containing the SID of the user, the Service's URL and the access request. The SID is used to identify the user and to prove that the user has accessed the SynService within the actual session. The URL of the SynService is used to authenticate the SynService against the UMService and to ensure that the response from the UMService is sent to the correct URL.

Afterwards, to prove that the Security Object is created by the authorized SynService, the service signs the Security Object with its private key. Furthermore, to prevent the SID being hijacked by a man-in-the-middle attack, the SynService encrypts the signed Security Object with the public key of the UMService. Therefore, only the UMService is able to decrypt the Security Object and check if the signature, the URL of

the SynService and the SID are valid. If the Object is identified as valid its request is processed. If further interaction of the user is required for processing the request, the SynService and the request are displayed to the user who can decide if the operations should be denied or permitted.

UMService access by a PService If a PService needs to access the user profile, the problem that the user does not know the PService directly, occurs. That leads to the situation, that the user is not able to determine if the PService has been invoked from a SynService and hence should be authorized to access the user profile or if the PService has sent the request without any invocation from a SynService. That means, that the user is not able to decide for a given PService if the PService should be authorized to access the user profile or not. The user, and hence the UMSERVICE, require the additional information which SynService has invoked the unknown PService. This is realized by the enhanced Security Object (figure 5).

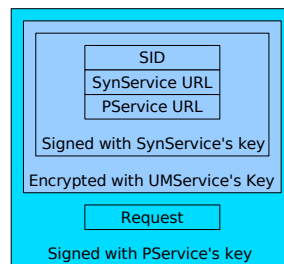


Fig. 5. The enhanced Security Object for SynServices

The SynService, that generates the inner part (light blue) of the enhanced Security Object now adds the URL of the PService that it invokes and submits the request. This inner part is signed and encrypted in the same way like the simple Security Object and afterwards, sent to the PService that needs to access the UMSERVICE. The PService now adds its request and signs it with its private key. This package is sent to the UMSERVICE.

The UMSERVICE now decrypts the inner part, checks if the URL of the PService is valid (does the signature of the Security Object match to the public key, given for the inner PService URL?). Further checks are performed in the same way as for the simple Security Object. If user interaction is required while the request is processed, the user is now supplied with the additional information about which SynService has invoked the PService.

5 Scenario and Evaluation

In this section we compare our approach, the User Modelling Service, with the state-of-the-art approaches User Modelling Server and per-application user modelling. Therefore, we use applications from within the E-Learning domain.

5.1 Semantic Web E-Learning

To compare the different approaches, we use three different applications: The C++ E-Learning Application, the Java E-Learning Application, and the Semantic Web Course E-Learning Application.

Java E-Learning Application The Java E-Learning Application¹⁰ is known to all systems. It has an overlap with the C++ E-Learning Application in some generic programming concepts, like +if+ or +case+ directive and object oriented programming concepts.

Semantic Web Course E-Learning Application The Semantic Web Course is known to all systems. It has no overlap with other E-Learning Application.

C++ E-Learning Application The C++ E-Learning Application is unknown to both centralized User Modelling approaches as it is new. But it uses an Ontology that references to the already known Ontology of the Java E-Learning Service.

5.2 Evaluation

We evaluate the approaches by means of the following criterias:

- How can the approach handle the situation that new services or applications appear?
- Can the generated user profile be shared with other applications?
- Can the data be accessed by an unauthorized third party?
- Has the user full control of his data (cf. can he delete, modify, and restrict access to the data)?

per-application user modelling As the per-application user modelling approach has to be re-implemented for every application it is a task of the application's developer and therefore should be working as soon as a new application appears, the same applies for the centralized domain knowledge: The application's programmer has to create an own Ontology or uses an existing one. There is no need for maintaining a centralized Ontology. As the user modelling process and the storage of the User Profile data are combined at the application's side the user has only limited possibilities to modify the data and has no possibility to control the access to his data. The per-application user modelling approach does not enable reuse of data as there exists no interface that enables other applications to access user profile data.

¹⁰ <http://www.sis.pitt.edu/~taler/QuizGuide.html>

User Model Server The User Modelling Server enables cross application usage of User Profile data as it stores the User Profile centralized. The profile is generated from events that are evaluated from the User Modelling Server, hence it needs to have domain knowledge and requires an update if a new Service or application appears. With an appropriate implementation, the User Modelling Server gives the user full control on his data and on the access control.

User Modelling Service The User Modelling Service stores the User Profile in a centralized place and therefore features cross application usage of the User Profile. As the modelling task itself is performed on the application side there is no need for a centralized Ontology or an update of a centralized component if new applications emerge. Furthermore, new applications can immediately access the User Profile and store their own data in it.

Summary In summary the three approaches compare as follows: per-application user modeling scales well, but the data is kept away from the user. User Modelling Servers need updates of the centralized components if new Web Services emerge. The User Modelling Service combines the advantages of both approaches:

Table 1. Comparison of different user modelling approaches

	per-application UM	UM Server	UM Service
New Services emerge	+	-	+
Reusage of User profiles (Cross application)	-	+	+
Security of data	-	+	+
Centralized domain knowledge	+	-	+
User can control data	-	+	+

6 Conclusion and Further Work

In this paper we presented the User Modelling Service, which can be used as a centralized tool for storing an aggregated User Profiles across several applications. It combines the advantages of the per-application user modelling and the User Model Server approach: The UMService is scalable in terms of the appearance of new Web Services and new domain and enables a cross application usage of the user profile while preserving domain aware user modelling. Furthermore, it protects the privacy of the users: The user can at every point view and control his own user profile and can define access rules about properties that can be accessed and modified by certain services.

7 Acknowledgement

This research has partially been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. <http://rewerse.net>).

References

1. Baldoni, M., Baroglio, C., Brunkhorst, I., Marengo, E., Patti, V.: A personalization service for curriculum planning. In: ABIS 2006 - 14th Workshop on Adaptivity and User Modeling in Interactive Systems. (October 2006)
2. Kobsa, A.: Generic user modeling systems. *User Modeling and User-Adapted Interaction* 11(1-2) (2001) 49–63
3. Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., von Wilamowitz-Moellendorff, M.: Gumo - the general user model ontology. In: Proceedings of the 10th International Conference on User Modeling, Edinburgh, UK, LNAI 3538: Springer, Berlin Heidelberg (2005) 428–432
4. Heckmann, D., Krüger, A.: A user modeling markup language (userml) for ubiquitous computing. In Brusilovsky, P., Corbett, A.T., de Rosis, F., eds.: *User Modeling*. Volume 2702 of *Lecture Notes in Computer Science*., Springer (2003) 393–397
5. Berkovsky, S., Kuflik, T., Ricci, F.: Cross-technique mediation of user models. In Wade, V.P., Ashman, H., Smyth, B., eds.: *AH*. Volume 4018 of *Lecture Notes in Computer Science*., Springer (2006) 21–30
6. Abel, F., Baumgartner, R., Brooks, A., Enzi, C., Gottlob, G., Henze, N., Herzog, M., Kriesell, M., Nejd, W., Tomaszewski, K.: The personal publication reader, semantic web challenge 2005. In: 4th International Semantic Web Conference. (nov 2005)
7. Henze, N., Kriesell, M.: 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)
8. Henze, N., Krause, D.: Personalized access to web services in the semantic web. In: SWUI 2006 - 3rd International Semantic Web User Interaction Workshop, Athens, Georgia, USA (nov 2006)
9. Bonatti, P.A., Duma, C., Fuchs, N., Nejd, W., Olmedilla, D., Peer, J., Shahmehri, N.: Semantic web policies - a discussion of requirements and research issues. In: 3rd European Semantic Web Conference (ESWC). Volume 4011 of *Lecture Notes in Computer Science*., Budva, Montenegro, Springer (June 2006)
10. Heckmann, D.: *Ubiquitous User Modeling*. PhD thesis, Department of Computer Science, Saarland University, Germany (November 2005)