A Framework for Flexible User Profile Mashups

Fabian Abel[1], Dominikus Heckmann[2], Eelco Herder[1], Jan Hidders[3], Geert-Jan Houben[3], Daniel Krause[1], Erwin Leonardi[3], Kees van der Slujis[4]

[1] L3S Research Center, Leibniz University Hannover, Germany
{abel,herder,krause}@l3s.de
[2] DFKI GmbH, Saarbrücken, Germany heckmann@dfki.de
[3] Web Information Systems, TU Delft, The Netherlands
{a.j.h.hidders,g.j.p.m.houben,e.leonardi}@tudelft.nl
[4] Department of Computer Science, Eindhoven University of Technology, The Netherlands k.a.m.sluijs@tue.nl

Abstract. Exploiting the rich traces of users' Web interaction promises to enable cross-application user modeling techniques, which is in particular interesting for applications that have a small user population or that are used infrequently. In this paper we present a framework for the effective interchange of user profiles. In addition to derivation rules for user profile reasoning, the framework employs flexible mash-ups of RSS-based user data streams for combining heterogeneous user data in a Web 2.0 environment.

1 Introduction

With the increased use of search engines, e-commerce systems and social networking sites – with famous examples such as Amazon, Facebook, Flickr, Delicious and Google – user modeling and Web personalization has evolved from a rather marginal activity to a mature technology that is exposed to the majority of Web users on a daily basis. Most techniques are based on collaborative filtering and social network analysis [1]. What they have in common is that they are rather straightforward and depend on a sufficiently large number of users that regularly interact with the system [2].

Apart from the major players in the field, many systems cannot boast on a large user base. These systems vary from startups to well-established sites that serve a specialized audience. As an example, e-learning systems inherently have a limited audience, in particular if the system is specifically used by one institution. For these stakeholders, it would be beneficial to have user profile information from other applications. Recent research suggests that, if carefully designed and tested, heterogeneous types of data can be used for reliably classifying users [3, ?]. Other motivations for cross-application user modeling include the synchronization of recommendations and user interaction between applications and better support of user migration.

Obviously, the idea of cross-application user modeling is not new. In the 1990s several *generic user modeling servers* have been developed, to be used by a wide range of applications (for example [4]). One of the major reasons that this approach has never been successful is that these servers were centralized, making use of predefined structures. By contrast, user models differ significantly between applications, depending on the adaptation goals, the context of use, privacy concerns, the design philosophy and many other factors.

New trends from the Web 2.0 as well as the related work, as will be discussed in Section 2, motivate an infrastructure for cross-application user modeling. This infrastructure, which we introduce in Section 3, is heavily inspired by social networking approaches and is based on the assumption that adaptive systems (or rather the system administrators) themselves are the ones who know best what the system needs. The infrastructure relies on the brokerage of user models, with system administrators searching, discussing, adopting, rating and recommending third parties' user models. Section 4 outlines how to use the framework to reason on distributed user profiles and demonstrates how user profiles can be mashed-up by combining RSS feeds in so-called *user pipes*.

2 Background and Related Work

As described in Tim O'Reilly's Web 2.0 design patterns [5], small sites with a small user population and specific demands make up the bulk of the Web 2.0 domain. Whereas the exchange of login credentials is already facilitated by initiatives such as OpenID[1], still in most cases users need to build their user profiles from scratch for every application. A recent trend is the combination of functionality from multiple Web 2.0 applications in so-called *mashups*. For mashups, the ability to share user profiles is particularly essential for a better integration and cooperation between the single applications.

For the exchange and interpretation of user profile data, common semantics user profile statements are needed [6]. Possible formats for user profiles include the General User Model Ontology (GUMO) [7] or Friend of a Friend (FOAF) [8]. However, as we have seen in the introduction of this paper, these kinds of predefined and static user profile ontologies do not sufficiently cater for the diverse needs of applications. Therefore, we argue that these types of shared models should rather be built bottom-up, starting from successful implementations in specific systems [9].

As a further development, we can see a shift from author-predefined adaptation rules to collaborative filtering techniques and the use of Web 2.0 interaction mechanisms [10]. With a huge pool of data, many candidate user groups to compare the user with, and several methods at hand, it becomes even more important to experiment with and optimize the conceptual adaptation decisions [11].

In essence, there are two ways to ensure interoperability between two adaptive systems and their user models. The first approach involves a *lingua franca*, an agreement between all parties on a common representation and semantics [12]. As described in the introduction, this is the philosophy underlying the generic user model server approach, used by CUMULATE [13] or PersonIs [14]. Given the wide variety in system objectives and the associated user models, generic

[1] http://openid.net/

user model servers have never gained wide acceptance. An alternative approach, which is more flexible, involves *conversion* between the different systems' user models.

Conversion allows for flexible and extensible user models, and for systems to join into a platform. Moreover, in contrast to a fixed lingua franca approach, conversion is suitable for 'open-world user modeling', which is not restricted to one specific set of systems [15]. This flexibility comes at a price, though. In addition to possibly losing information in the conversion process, it might be that models are simple incompatible (in the sense that there is no suitable mapping) or that mappings are incomplete (information required in one model is not available in the other). Given that there are suitable mappings, the observations in the different systems may lead to contradictions [15]. Several methods for conflict detection and resolution are conceivable, among others reliability weighting and majority voting - again, which method to use, may be a subjective design decision.

As pointed out by [16], computer-based representation of *provenance data* is crucial for users who want to analyze, reason, and decide whether or not they trust electronic data. In the article, the generic concept of p-statements is explained: each statement should contain a track record of the input data, the processing and a description of the output data. With this information, a derivation record can be built for analysis purposes. The DCMI Metadata Terms [17] is a collection of properties and classes together with vocabulary and syntax encoding schemes that can be applied to describe the provenance of data as well. The DCMI terms allow to describe metadata of things, such as the creator, time of creation, copyright and modifications.

3 A Framework for User Modeling 2.0

Results from the preceding section provide support for the exchange of user models between applications. From the related work we have seen that incorporating user profile information from other contexts is not a straightforward process, though. The poor take-up of the generic user modeling servers, developed in the 1990s, suggests that a centralized approach, with predefined ontologies, does not cater the needs of the multitude of adaptive systems, which are very heterogeneous in nature.

Based on the above, we designed a framework that facilitates the *brokerage* of user profile information and user model representations. This framework, which we call the Grapple User Modeling Framework (GUMF), is designed to meet the following requirements. First, various types of systems should be able to connect to the framework. Further, the framework should provide a *flexible user model format* that allows for new types of statements and derivation rules. Sufficient *metadata* should be given to indicate its origin, contents and validity. The browsing and searching of user data or model extensions, provided by the connected systems, should be supported by *rating mechanisms.* As several systems may provide *competing* models of, for example, user interests, and as the quality of these models can vary significantly it is important that a system

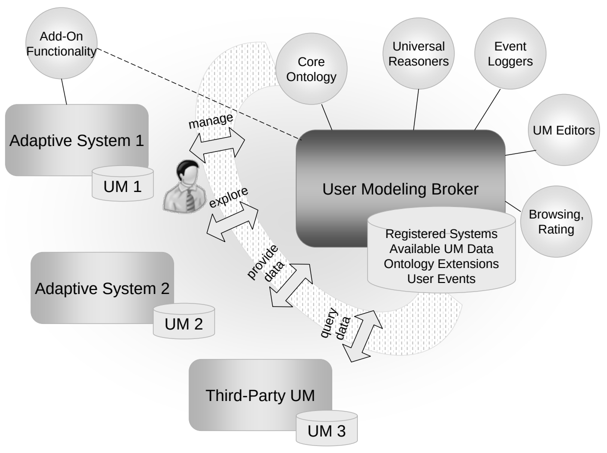


Fig. 1. Generic overview of the functionality of the User Modeling Framework.

administrator (i.e. a user of the framework) can take a motivated decision which alternative is most suitable for his personalization purposes.

The core element of the framework can be considered a *broker*, which provides the means for other systems to share and make use of their user data. In this section we provide an overview of the elements that are needed for setting up this framework.

3.1 Architecture

In Figure 1, a generic overview of the GUMF architecture is depicted. The central element of the framework is the Grapple User Modeling Broker (GUMB), which manages the communication between the connected systems. The broker keeps track of the registered systems, the available user model data and ontology extensions. Further, it keeps a centralized repository of user events. The framework provides Web-based administrative interfaces for *managing* the system configuration and for *exploring* the available user data streams, reasoning mechanisms and ontology extensions. The target audience of these interfaces consists of the administrators and programmers of client (adaptive) systems, in order to find and incorporate suitable user data streams and to offer their own data streams. For most mapping, merging and reasoning tasks, administrators can utilize generic reasoning plugins (cf. Section 4) and hence generate user profile data in a format that perfectly fit their applications' needs. For more specific reasoning tasks, administrator can create own reasoning plugins an provide them to the GUMF community. Once configured, the client systems can exchange user data without human intervention. The *provision* of data takes place in the form of statements, of which the structure is explained in more detail in Section 3.2.

The *querying* of user data – summarized in statements – is realized through three alternative interfaces. The RESTful interface provides a light-weight querying approach for retrieving statements that match a certain simple pattern. A more elaborate interface is provided by a SOAP interface, which is more flexible,

property	**description**
ID	The globally unique ID of the statement.
type	In the current version of the UM ontology we differentiate between *gc:Statement*, which is a basic user profile statement, and *gc:Observation*, which is a specialization of *gc:Statement* and models a (user) observation made in some application.
subject	The entity (usually the user) the statement is about.
predicate	Refers to a property (of a domain ontology) that either characterizes the *subject* (e.g. *foaf:interest* or *k:hasKnowledge*) or describes some action the *subject* has performed (e.g. *nop:hasBookmarked* or *nop:hasClicked*).
object	The value of the *predicate* (e.g. *"ItalianFood"* or *dbpedia:semantic_web*).
created	Specifies when the statement was created.
creator	Refers to the entity that created the statement. In case of a *gc:Observation* it identifies the entity that reported the observation.
temporal	Allows to define temporal constraints on the validity of the statement.
evidence	If a statement was produced by a reasoning process then *evidence* can be used to show how the statement was deduced.
rating	The *rating* of a statement indicates the level of trust in the statement.

Table 1. Important properties of a Grapple statement as defined in the Grapple User Modeling Ontology (see: http://www.kbs.uni-hannover.de/gumf.owl).

at the cost of a more complicated syntax and communication costs. A third interface allows applications to subscribe to an RSS-based data stream that matches a query, to be notified upon changes. The latter interface is particularly useful for event-driven personalization mechanisms, which depend on events in other systems.

The GUMF architecture is inspired by the Personal Reader Framework [18], with as main enhancements the extensible user modeling ontology format, flexible query interfaces and a community-based way of sharing and ranking user models.

3.2 User Modeling Ontology

The Grapple User Modeling Ontology specifies the lingua franca for exchanging user profile information and user observations in a User Modeling 2.0 infrastructure. It follows the approach of the General User Model Ontology [7] (GUMO) and UserRDF [19], as it is built upon the notion of reified *subject-predicate-object* statements. The *subject* models the entity (usually the user) that the statement is about. The *predicate* refers to a property that either characterizes the subject (e.g. *foaf:interest* or *k:hasKnowledge*) or describes some action the subject has performed (e.g. *nop:hasBookmarked* or *nop:hasClicked*). The *object* contains the corresponding value (e.g. *"ItalianFood"* or *dbpedia:semantic_web*). Each statement has a globally unique ID and is enriched with metadata (see Table 1), such as the creation date or details about the provenance of the statement.

```
gc   = http://www.grapple-project.org/grapple-core/
foaf = http://xmlns.com/foaf/0.1/
gc:Statement {
  gc:id:        gc:statement-peter-2009-01-01-3234190;
  gc:user:      http://www.peter.de/foaf.rdf#me;
  gc:predicate: foaf:interest;
  gc:object:    http://en.wikipedia.org/wiki/Italy;
 }
```

In the example above, the subject (gc:user), predicate, and object refer to entities that are not part of the Grapple Core ontology. gc:user identifies the user Peter by referring to his FOAF profile, which is a separate document located at "http://www.peter.de/foaf.rdf". The value of the predicate is "foaf:interest"., which is a property defined in the FOAF ontology [8]. To find out about the actual meaning of "foaf:interest", one has to look up the FOAF ontology[2]:

```
<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/interest"
     vs:term_status="testing"
     rdfs:label="interest"
     rdfs:comment="A page about a topic of interest to this person.">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
  <rdfs:range rdf:resource="http://xmlns.com/foaf/0.1/Document"/>
  <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/>
</rdf:Property>
```

The definition of "foaf:interest" gives us the actual meaning of the Grapple statement. The comment describes the semantics of the predicate, to be read by people that want to use the property. Making use of the definitions of the domain and range, we can deduce that "http://www.peter.de/foaf.rdf#me" is of the type "foaf:Person", that "http://en.wikipedia.org/wiki/Italy" is a "foaf:Document" and that the predicate "foaf:interest" reflects 'A page about a topic of interest to this person'.

4 User Profile Reasoning

The Grapple User Modeling Framework allows to dynamically utilize reasoning plugins to enable user profile reasoning. In this section we present two generic solutions that can be utilized directly by the GUMF client applications: (1) a rather classical rule-based approach and (2) a novel approach, which we call *User Pipes*, that allows user profile reasoning by mashing up different user profile data streams. However, client administrators can also create own reasoning plugins and share them with the community. A user interface within the client administrator backend allows to search for and publish own reasoning plugins.

4.1 Reasoning Plugins

Reasoning plugins are software components that can be integrated into the Grapple User Modeling Framework (GUMF). In general, they deduce new information about a user based on existing user profile data or based on some observations. Reasoning plugins can come in different flavors. For example, a plugin might gather and align user data from different social networking services in order to create a more comprehensive user profile.

The first generic reasoning plugin is rule-based and applies derivation rules, which can be defined and adjusted by client applications. These derivation rules enable GUMF to generate new Grapple statements. Rules allow to express simple

[2] More precisely, the ontology that is identified via foaf = http://xmlns.com/foaf/0.1/

types of inference in terms of premise-conclusion rules that derive new statements from the existence of other statements. These rules can, for example, (i) infer statements that embody new knowledge, (ii) they can be used to map between different ontologies or (iii) they describe how to solve problems where statements or rules conflict with each other. A simple derivation rule that infers new knowledge about a user might express the following: If a user has bookmarked a website that has topic t then the user is interested in t. Such a rule can, for example, simply be formulated as a SPARQL query:

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX gc:    <http://www.grapple-project.org/grapple-core/>
PREFIX gnop:   <http://www.grapple-project.org/nop/>

CONSTRUCT { gc:derivedStatement  gc:user      ?user .
            gc:derivedStatement  gc:predicate foaf:interest .
            gc:derivedStatement  gc:object    ?topic }
WHERE {
            ?originalStatement   gc:user      ?user
            ?originalStatement   gc:predicate gnop:hasBookmarked .
            ?originalStatement   gc:object    ?document .
            ?document            foaf:topic   ?topic . }
```

A mapping rule could simply map one value to another value or it can compose a new value from other values or decompose one value in different separate values. Conflict resolution rules can be used to define preferences among different types of statements or preferences among different rules.

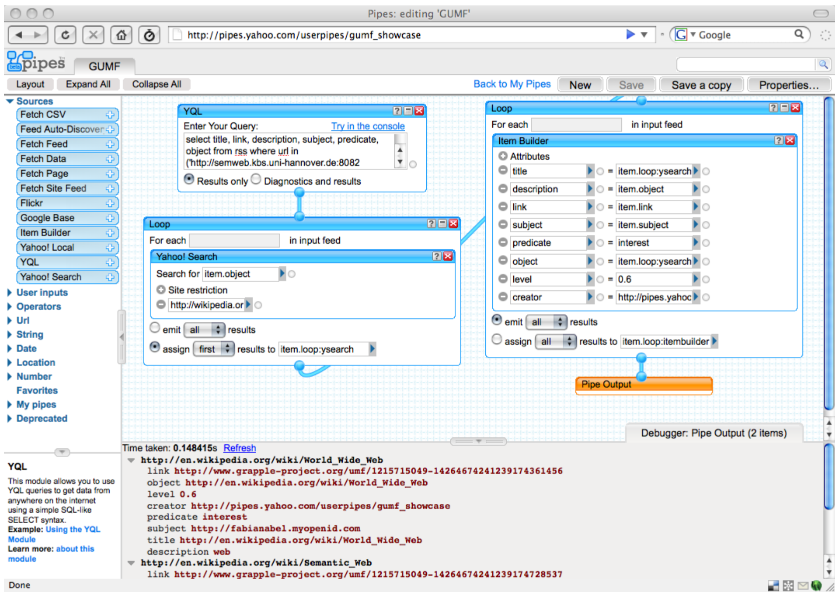
4.2 User Pipes

In addition to the rule-based approach described in the section above, GUMF enables deduction of user profiles also by mashing up different (user profile) data streams in RDF or RSS-format by utilizing Semantic Web Pipes[3] or Yahoo Pipes[4]. In this chapter, we focus on the processing of RSS data by utilizing Yahoo pipes as this enables the usage of a huge amount of structured data on the web. Different RSS streams are syndicated to so-called *User Pipes*.

How this works is shown by our GUMF demonstrator[5]. A specific profile stream *searchedFor* of the user *fabian* can be retrieved by requesting */user/fabian/predicate/searchedFor*. An extract of the data stream is given as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF ...>
  <channel rdf:about="http://semweb.kbs.uni-hannover.de:8082/grapple-umf/user/fabian">
    <title>GUMF data stream matching the query 'user = fabian'</title>
    <link>http://semweb.kbs.uni-hannover.de:8082/grapple-umf/user/fabian</link>
    <items>
      <rdf:Seq>
        <rdf:li rdf:resource="http://semweb.kbs.uni-hannover.de:8082/grapple-umf/62715"/>
        <rdf:li rdf:resource="http://semweb.kbs.uni-hannover.de:8082/grapple-umf/63526"/>
        ...
      </rdf:Seq>
    </items>
  </channel>
```

[3] http://pipes.deri.org/
[4] http://pipes.yahoo.com
[5] Available at http://semweb.kbs.uni-hannover.de:8082/grapple-umf/



YQL

This module allows you to use YQL queries to get data from anywhere on the internet using a simple SQL-like SELECT syntax.
Example: Using the YQL Module
Learn more: about this module

YQL — Enter Your Query: Try in the console

```
select title, link, description, subject, predicate,
object from rss where url in
('http://semweb.kbs.uni-hannover.de:8082
```

[✓ Results only] Diagnostics and results

Loop — For each in input feed
Yahoo! Search — Search for item.object; Site restriction: http://wikipedia.or
emit all results [✓ assign first results to item.loop:ysearch]

Loop — For each in input feed
Item Builder — Attributes

title	item.loop:ysearch
description	item.object
link	item.link
subject	item.subject
predicate	interest
object	item.loop:ysearch
level	0.6
creator	http://pipes.yahoo

[✓ emit all results] assign all results to item.loop:itembuilder

Time taken: 0.148415s Refresh

```
▼ http://en.wikipedia.org/wiki/World_Wide_Web
  link http://www.grapple-project.org/umf/1215715049-14264674241239174361456
  object http://en.wikipedia.org/wiki/World_Wide_Web
  level 0.6
  creator http://pipes.yahoo.com/userpipes/gumf_showcase
  predicate interest
  subject http://fabianabel.myopenid.com
  title http://en.wikipedia.org/wiki/World_Wide_Web
  description web
▼ http://en.wikipedia.org/wiki/Semantic_Web
  link http://www.grapple-project.org/umf/1215715049-14264674241239174728537
```

Fig. 2. User Pipe: Mashing up user profile data streams from different sources (here: GUMF search activity stream and Delicious bookmarks). Online available at: http://pipes.yahoo.com/userpipes/gumf_showcase

```
  <item rdf:about="http://semweb.kbs.uni-hannover.de:8082/grapple-umf/62715">
    <title>user 'fabian' searched for 'Trento'</title>
    <link>http://semweb.kbs.uni-hannover.de:8082/grapple-umf/62715</link>
    <gc:subject>http://fabian.myopenid.com</gc:subject>
    <gc:predicate>http://www.grapple-project.org/nop.owl#searchedFor</gc:predicate>
    <gc:object>Trento</gc:object>
    <gc:level>1.0</gc:level>
    <gc:created>2009-03-20T18:23:50Z</gc:created>
    <gc:creator>http://bookstore.example.org</gc:creator>
    ...
  </item>
  ...
</rdf:RDF>
```

This data stream can be combined with other data streams to deduce new user profile information. For example, it can be combined with information from the feed */user/fabian/predicate/interest* to deduce whether the user's interests and search activities are thematically similar or it can even be mashed up with other RSS feeds from the Web.

To demonstrate how meaningful streams can be created by embedding profile data from social networking sites, we created a simple user pipe[6] that combines the search activity stream listed above with the latest bookmarks that the user created at Delicious[7]. Figure 2 shows the editor view of the user pipe. The given user pipe detects those keywords that a user applied for both search and tagging of his latest bookmarks, which is expressed via the following YQL query.

```
SELECT title, link, description, subject, predicate, object FROM rss WHERE url in
```

[6] Available at http://pipes.yahoo.com/userpipes/gumf_showcase

[7] http://feeds.delicious.com/v2/rss/fabianabel

```
  ('http://semweb.kbs.uni-hannover.de:8082/grapple-umf/user/fabian/predicate/searchedFor')
 AND object in
  (select category from rss where url in ('http://feeds.delicious.com/v2/rss/fabianabel') )
```

The result of the YQL query is then passed to a component that tries to map the detected keywords to Wikipedia articles that further explain the concepts that are referred by the keywords. In the last stage, an *Item Builder* component is used to generate new Grapple statements. Similar to the example in Section 3.2, the above item makes use of the FOAF vocabulary (*foaf:interest*) to express that the user is interested in *http://en.wikipedia.org/wiki/Trento* (cf. bottom of Fig. 2):

```
<?xml version="1.0" encoding="UTF-8"?>
  ...
  <item rdf:about="http://www.grapple-project.org/umf/1215715049-14264674241239174361456">
    <title>http://en.wikipedia.org/wiki/Trento</title>
    <gc:subject>http://fabian.myopenid.com</gc:subject>
    <gc:predicate>http://xmlns.com/foaf/0.1/interest</gc:predicate>
    <gc:object>http://en.wikipedia.org/wiki/Trento</gc:object>
    <gc:creator>http://pipes.yahoo.com/userpipes/gumf_showcase</gc:creator>
  </item>
  ...
```

The benefit of the user pipe approach is that user pipes result in user profile streams that can again be used by other profile reasoners, which allows for flexible and extensible user profile reasoning. For publicly available data streams it is also possible to directly use the Yahoo Pipe editor, which provides an easy drag-and-drop user interface to process, combine, and perform various operations on data streams. This means that not only programmers or experts familiar with SPARQL or rule-based languages are enabled to create profile reasoners, but also leisure user as they can create such reasoners (user pipes) visually.

The critical point of this approach is the immensely huge amount of RSS data on the Web that could slow down the processing of a pipe. Therefore, we are going to explore caching strategies (e.g. the precompute pipes regulary and deliever the cached results) as proposed in [13] and will conduct performance measures as well.

5 Conclusions and Future Work

In this paper we motivated and introduced a framework for cross-application user modeling. Based on several pieces of earlier work, the framework provides a domain-independent, decentralized approach for combining several user models. In a collaborative manner, the connected systems can create, share, select, mashup, adopt and rate their user models, supported by a basic infrastructure that includes search and browse facilities, editors and universal reasoning mechanisms.

Although the framework provides the basic infrastructure for cross-application modeling, its success depends on the take-up by a critical mass and the availability of the necessary tools. In the GRAPPLE project, we are currently integrating the framework, to be used by a number of different e-learning systems. By evaluation and experimentation, we expect to find additional requirements and success factors for building an ecology of adaptive systems that exchange parts of their user models.

Acknowledgements The work presented in this paper has been sponsored by the EU FP7 STREP Project GRAPPLE.

References

1. Frias-Martinez, E., Magoulas, G., Chen, S., Macredie, R.: Modeling human behavior in user-adaptive systems: Recent advances using soft computing techniques. Expert Systems with Applications **29** (2005) 320–229
2. Bilenko, M., White, R.: Mining the search trails of surfing crowds: Identifying relevant websites from user activity. In: Proc. WWW2008. (2008)
3. Korth, A., Plumbaum, T.: A framework for ubiquitous user modeling. In: Proc. Information Reuse and Integration 2007. (2007)
4. Kobsa, A., Koenemann, J., Pohl, W.: Personalized hypermedia presentation techniques for improving customer relationships. The Knowledge Engineering Review **16 (2)** (2001) 111–155
5. Oreilly, T.: What is web 2.0: Design patterns and business models for the next generation of software. Social Science Research Network Working Paper Series (2007)
6. Ankolekar, A., Krötzscha, M., Trana, T., Vrandecic, D.: The two cultures: Mashing up web 2.0 and the semantic web. In: Web Semantics: Science, Services and Agents on the World Wide Web. (2008)
7. Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., von Wilamowitz-Moellendorff, M.: Gumo - the general user model ontology. In: Proc. of the 10th Int. Conf. on User Modeling, Edinburgh, UK (2005) 428–432
8. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.91. Namespace document, FOAF Project (November 2007) http://xmlns.com/foaf/0.1/.
9. Paramythis, A., Loidl-Reisinger, S.: Adaptive learning environments and e-learning standards. Electronic J. of e-Learning **2 (2)** (2004) Paper 11
10. Chatti, M.A., Jarke, M.: The future of e-learning: a shift to knowledge networking and social software. Int. J. Knowledge and Learning **3 (4/5)** (2007) 404–420
11. Nielsen, J.: Personalization is over-rated. Alertbox (October 1998)
12. Stewart, C., Celik, I., Cristea, A., Ashman, H.: Interoperability between aeh user models. In: Proc. APS 2006. (2006)
13. Yudelson, M., Brusilovsky, P., Zadorozhny, V.: A user modeling server for contemporary adaptive hypermedia: An evaluation of the push approach to evidence propagation. In: 11th International Conference on User Modeling. (2007) 27–36
14. Assad, M., Carmichael, D., Kay, J., Kummerfeld, B.: Personisad: Distributed, active, scrutable model framework for context-aware services. (2007) 55–72
15. Aroyo, L., Dolog, P., Houben, G., Kravcik, M., Naeve, A., Nilsson, M., Wild, F.: Interoperability in pesonalized adaptive learning. J. Educational Technology & Society **9 (2)** (2006) 4–18
16. Moreau, L., Groth, P., Miles, S., Vazques-Salceda, J., Ibbotson, J., Jiang, S., Munroe, S., Rana, O., Schreiber, A., Tan, V., Varga, L.: The provenance of electronic data. Communications of the ACM **51 (4)** (2008)
17. DCMI Usage Board: DCMI Metadata Terms. DCMI Recommendation, Dublin Core Metadata Initiative (January 2008)
18. 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)
19. Abel, F., Henze, N., Krause, D., Plapper, D.: User modeling and user profile exchange for semantic web applications. In: 16th Workshop on Adaptivity and User Modeling in Interactive Systems, Wuerzburg, Germany. (2008)