Ontology-Based Multidimensional Personalization Modeling for the Automatic Generation of Mashups in Next-Generation Portals

Fedor Bakalov and Birgitta König-Ries
Department of Computer Science, University of Jena
Ernst-Abbe-Platz 1-4, 07743 Jena, Germany
{fedor.bakalov|koenig}@informatik.uni-jena.de

Andreas Nauerz and Martin Welsch
IBM Research and Development
Schönaicher Str. 220, 71032 Böblingen, Germany
{andreas.nauerz|martin.welsch}@de.ibm.com

Abstract

Recently, the combination of portal and mashup technology has gained some attention. Portals were originally designed as single points of access to the information and applications distributed across the enterprise. However, due to the increasing number of resources available through portals, they have gained a new challenging goal: To provide users with the information tailored to their individual needs and geared to the situation they are working in. Mashups, the tools that dynamically integrate independent applications, seem to be a good technique to achieve this goal. What is needed, however, are means to automatically create personalized mashups that optimally fit a user's information needs in a given situation. In this paper, we describe our approach to this automatic mashup generation. At the core of our approach are different ontology-based models that describe the user, the domain, possible information needs in this domain, and personalization rules determining which information to present to which user in which situation.

1. Introduction

Portals were originally conceived as single points of access to all the information an enterprise user might need. Over time, however, it turned out that this is an unrealistically high aim to reach for: Most companies today run several portals and enterprise users are interested not only in the information provided internally, but also in the information offered by external providers. Consider for instance a finance manager reading a news bulletin about rumors of a planned merger of her company's competitor with another company and the consequences this has for the stock market. In this situation, the manager might be interested in a number of different additional pieces of information: the situation on the stock market, including the stock quotes of the companies mentioned in the text, an executive summary of the technology mentioned in the article and the list of people from her department who are familiar with this technology. Part of this information will be provided by the portal she is already using (but not necessarily right next to the information she is looking at right now), other parts of this information may well be spread over different other portals. In order to fulfill her information needs, a dynamic aggregation of all of this information is necessary.

The most suitable way to achieve this aggregation seems to be the combination of portal and mashup technology[1]. One of the most concise definitions of mashups is given by Lerner in [11]: "A mashup is a combination of two or more Web APIs in a novel way, making information more accessible and informative than it would be on its own". Today, a number of tools allow for the manual creation of mashups. These tools are suitable for users with some programming skills and an interest in "playing around" with the technology. However, they are not suitable for the average user who is just interested in obtaining the right information at the right time. With current solutions, these users are relegated to using existing mashups that someone has created and made available. This will only seldom result in having them available when one needs them.

Therefore, in this paper, we take mashup generation one step further: We will discuss how an automatic generation

[1] see for instance http://blogs.zdnet.com/BTL/?p=4912 for a summary of Gartner report on this topic.

of appropriate mashups can be supported. It will turn out that in order to achieve this goal, we need a number of ontology-based models: first of all, we need to describe the user, her interests and expertise. Second, we need to describe relevant domain concepts. Third, we need to identify which information gathering activities could be performed with respect to which of these concepts. Finally, we need to bring all three models together and define personalization rules that capture which information to offer to which user and when. In the remainder of the paper, we are going to discuss our framework and the individual models in more detail.

2. Related Work

In this section, we are first taking a look at existing systems – both commercial and from the research community – for mashup generation. Afterwards, we will introduce different approaches to user modeling and will discuss whether they are useful in our context.

2.1 Existing Systems for Mashup Generation

Several tools are available today for creating mashups. Such systems as Microsoft Popfly[2], Yahoo Pipes[3], and IBM's QEDWiki[4] provide users with a GUI environment where they can aggregate different data sources using certain operators and filters. They offer widgets associated with data sources (e.g. RSS feeds), which users can drag to the canvas and define how the data should be aggregated. However, these systems have a number of limitations. First, in order to create even a simple mashup, the user has to possess a certain level of technical knowledge. Second, mashups are mostly used to solve a short-term information need, for which users are not always willing to invest time manually assembling data. Third, the number of available data sources and aggregation operators is limited to what the system provides.

A number of mashup frameworks have been proposed to solve some of the problems mentioned above. Intel MashMaker is a tool implemented as a browser extension that analyzes the content the user is viewing and delivers additional information the user might need to accomplish her goal [6]. MashMaker extracts structured information from the Web pages using a collaboratively maintained database of extractors and uses this information to identify the sources of additional content that can be suggested to the user. The drawback of this approach is that it requires information about the parameterization of Web pages, which is manually provided by the user community. However, due to the dynamic nature of the Web it is difficult to maintain this parameterization information up-to-date.

In [5], Díaz et al. describe the MARGMASH tool that provides personalized mashups within the context of existing Web applications. The tool places so called "marginal mashups" along the pages of the application the user is working with. The content of these mashups is delivered through Yahoo Pipes, which are selected based on the information contained in the current page. In another approach, Thang et al. describe a prototype of a personalized travel assistant [15]. The software displays maps containing user-tailored information about weather, accommodation, dining places, etc.

DAMIA [14] is a rather comprehensive data integration platform that allows the creation, discovery, and management of quite complex mashups. It focusses on data integration issues in this context. It is geared towards manual assembly of mashups, though.

None of the existing systems, however, is able to dynamically and automatically create personalized mashups for portal users.

2.2 Related Approaches towards User Modeling

The user model is an essential component for any system that aims at adapting content to the users' specific needs. However, it is a challenging task to create an adequate user model that accurately represents user features. A number of proposals have been made to identify and represent knowledge about users. Crabtree and Soltysiak [4] describe an approach to deriving user interests automatically by monitoring various user activities, such as reading documents, writing emails, and browsing websites. The resulting user model is represented as a vector of weighted keywords denoting user interests. In [1], Achananuparp et al. describe how the vector user models can be semantically enhanced. They propose using WordNet[5] lexical database to establish the semantic relations between the keywords in the user model.

Ontologies have gained a large interest as a means to semantically represent knowledge about users. Semantic relations among concepts in the user model allow deriving certain information about users that was not explicitly defined in the model. For instance, ontological represetnation of user interests enables propagation of the interest from child concepts to their parents as well as among similar concepts. In [13], Schmidt describes ontology-based conceptual models that can be harnessed by learning management systems in order to provide users with the learning content tailored

[2] http://www.popfly.com/

[3] http://pipes.yahoo.com/pipes/

[4] http://services.alphaworks.ibm.com/qedwiki/

[5] http://wordnet.princeton.edu/

to the level of their knowledge as well as their situational needs. In [17], the authors elaborate an architectural solution that can automatically derive ontological user models based on Web content and user logs. A similar approach is described by Costa Pereira and Tettamanzi in [12].

An important aspect of user modeling is the ability to identify and represent the degree that the user is interested in a certain concept or possess knowledge on it, which in its turn can affect the quality of adaptation. In [9], Kavcic describes a novel approach to representing the degree of user knowledge using fuzzy set theory. John and Mooney describe a similar approach to represent user interests in [8].

From the analysis of related work, it is obvious that we need an ontology-based user model. The model needs to be able to capture user interests and expertise; for both an adoption of the fuzzy set based approaches seems to be suitable.

3. Mashup Framework

Our mashup framework provides users with personalized mashups automatically generated based on the knowledge about the user and the information about the context the user is acting in. The framework augments the documents that users are reading with background information and related content gathered from different sources and aggregated into one integrated tool. In the case of the finance manager mentioned in the introduction, for instance, the framework can display a side menu that shows the situation on the stock market, including the stock quotes of the companies mentioned in the text. Another side menu can be displayed to provide an executive summary of the technology mentioned in the article and the list of people from her department who are familiar with this technology.

Figure 1 illustrates the high-level components of the system architecture of our framework. In order to automatically augment the portal content with relevant information, we need a component that is able to extract machine-readable semantics from the content. For this purpose, we use the Calais[6] Web service and UIMA framework[7]. These analysis engines are able to extract certain types of entities, such as person, company, location, etc.

The semantics extracted from the content is used by the *personalization engine* to identify the information-gathering actions that provide the user the information she may need in a given situation. Each action is described by its input and output concepts, which denote the input data and wanted information respectively. The selection of actions is based on the personalization rules defined in the *personalization model* and the knowledge about the user interests and expertise stored in the *user model*, both of which are based on the *domain model*. More information on all four components will be provided in the following sections.

After the *personalization engine* has identified the information-gathering actions relevant to the user, the engine invokes a composition request for every of the selected actions. In the request, it specifies the input data and the information that must be obtained. Then the engine sends the request to the service composition module, the goal of which it is to find the services that provide the requested information. The composition module creates a composition (mashup model) by a multi-objective evolutionary algorithm [10], which operates on the semantic service descriptions provided by the *application registry*. The *application registry* maintains the references to the semantic and grounding service description for each of the registered RESTful or SOAPful services. After the composition, the module provides the generated mashup model to the *mashup handler*, which in turn invokes the *applications and services* and requests the *presentation module* to display the mashup on the portal page.

4. Domain Model

The key model in our framework that other models refer to is the *Domain Model*. A domain model is a structure that defines concepts and relations among them in a given domain, e.g., chemistry, medicine, biology, etc [7]. We have chosen the finance domain for our proof-of-concept implementation. Therefore, in our domain model we define the concepts that the users from the financial realm may work with, such concepts as stock, bank, account, etc.

With respect to the chosen domain, the basic goal of providing a mashup is to allow the user to gather additional information. Therefore, we need to model concepts related to the domain of information gathering, too. These concepts like *NewsArticle, EncyclopediaArticle, Map*, or *Expert* are also contained in the domain model. This part of the model is obviously independent of the application domain and can be reused.

The domain model is represented as an OWL ontology[8], which defines the domain concepts as ontological classes by specifying their properties and relations to other classes. E.g., an event *Acquisition*, denoting the fact of acquiring one company by another one, is defined as a subconcept of *FinancialTransaction* and described by such attributes as *date*, *acquiree*, *acquirer*, and *transationAmount*.

The ontology is grounded on the Proton upper-lever ontology[9], which defines upper-level concepts, such as organization, location, person, etc. Under these concepts, we define finance-specific terms partially reused from existing

[6] http://www.opencalais.com/

[7] http://incubator.apache.org/uima/

[8] http://www.minerva-portals.de/downloads/ontologies/finance-ontology.owl

[9] http://proton.semanticweb.org/

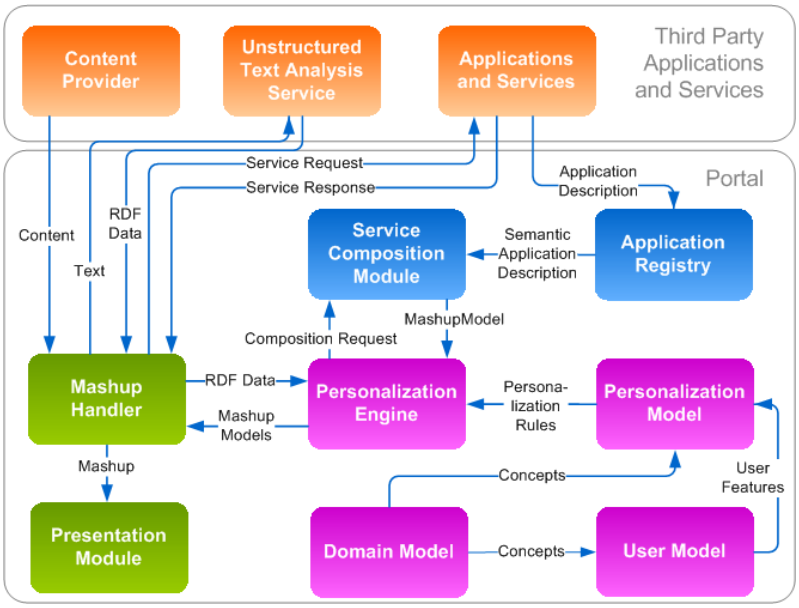

Figure 1. Mashup Framework

finance ontologies, namely LSDIS Finance Ontology[10] and XBRL Ontology[11].

In order to provide vocabulary to properly represent user interests and expertise in the user model, we define fine-grained categorization of the fields that belong or relate to the finance domain. These fields are represented as a taxonomy and defined under the *IndustrySector* concept. The categorization of fields is partially based on the Yahoo taxonomy[12].

The domain model also contains instances of concepts. E.g., for the concept *Company*, we specify such instances as 'Microsoft', 'IBM', 'Google', etc. Class instances are required to represent specific user interests in the user model. For example, our manager is reading a news article about an acquisition of *Company A* by *Company B*. The manager has been dealing with the former company for many years, hence she possesses expert knowledge on it. However, she has never heard anything about the latter company, therefore, she may need background information on it. In order to model such situation in the user model, we need to have instances of *Company A* and *Company B* explicitly defined in the domain model.

Inclusion of new instances into the ontology is performed automatically using the Calais service mentioned in Section 3. We harness the service in order to extract named entities (such as company, industry term, technology, etc) from the text of documents accessed by users. The extracted entities that are not yet present in the domain model are then inserted into the ontology as instances of the corresponding concepts.

5. Task Model

In addition to the definition of concepts and relationships between them, we need to conceptually define the actions that users might want to perform on the portal. We specify such actions in a *Task Model*. Here, we concentrate on the information-gathering activities that could be performed by the user on a certain object contained in the text she is reading. These activities are used in the *Personalization Model* to define the information that should be provided to the user when a certain object is present in the document. In its turn, the semantic description of these actions and their inputs and outputs is harnessed by the *Service Composition Module*, which goal is to generate composition of services that provide the required information.

The task model itself is represented as an OWL ontology, which defines the information-gathering actions as ontological concepts. Each action is described by input and output concepts. An input concept represents a set of input parameters that should be provided in order to perform a certain action; whereas an output concept denotes a set of output parameters that will be returned by the given action. We specify these parameters as object properties. For example, an input concept for *getEncyclopediaArticle* action has two object properties: *concept* and *userInterest*. The former property denotes the concept on which we want to obtain an encyclopedia article and it refers to an appropriate concept

[10] http://lsdis.cs.uga.edu/library/resources/
[11] http://xbrlontology.com/
[12] http://www.yahoo.com/

defined in the domain model, which is *EncyclopediaArticle*. The latter property is an optional parameter that is used to convey interests of a particular user (e.g., it could be harnessed by the service to extract a snippet of the article that matches the user interest); this property is of type *Entity*, which is the superconcept in our domain ontology.

In certain cases, we need to provide more detailed information on input and output parameters than just specifying their types. E.g., for *getCompanyAddress* action, we define *company* property of type *Company* in the input concept and *address* property of type *Address* in the output concept. However, this information alone is not sufficient for the *Service Composition Module* to automatically generate the right composition of services. We need a service that returns a company address, but not a person's address. We provide this additional semantics by putting a restriction on *address* property: $address \subset \forall belongsTo.Company$. OWL representation of such property restriction is given in Listing 1.

```
<owl:Class rdf:ID="getCompanyAddressOutput">
    <rdfs:subClassOf rdf:resource="#Output"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#address"/>
            <owl:allValuesFrom>
                <owl:Restriction>
                    <owl:onProperty
                     rdf:resource="#belongsTo"/>
                    <owl:allValuesFrom
                     rdf:resource="&dom;Company"/>
                </owl:Restriction>
            </owl:allValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Listing 1: Property Restriction Example

The task model contains two types of actions: general actions and domain-specific actions. The first type are those actions that could be used across different domains, e.g., *getEncyclopediaArticle*, *getNews*, *getPersonAddress*, etc. The domain-specific actions represent the activities that are applicable only in a certain field. As mentioned above, we concentrate on financial domain. Therefore, in our task model we define the information-gathering actions that are likely to be performed by the users working with financial information. Among others, the model defines such actions as *getStockQuotes*, *getCurrencyExchangeRates*, *getMarketStatistics*, etc.

6. User Model

In our framework, we aim to automatically generate personalized mashups that satisfy information needs of individual users. For this purpose, we construct a *User Model*, which reflects various user features, such as demographic characteristics, interests, and expertise. The model consists of two parts: static and dynamic. The static part contains more or less invariant information about users: age, gender, mother tongue, etc. This is the information that users provide explicitly when they register in the portal. In the dynamic part, we represent the constantly changing user characteristics, namely user interests and expertise.

We model interests and expertise as independent user features in order to provide the user only with relevant information, which is tailored to her specific needs and adjusted to the level of knowledge she has on a certain concept. This allows us to avoid the problem of, for example, displaying a Wikipedia article about Internet banking to an expert providing consulting services in the field of banking.

The user interests and expertise are represented as an overlay model [2]: They are defined as a subset over the domain ontology. In the overlay user model, we refer to concepts from the domain ontology and specify the degree of interest the user has in these concepts. Also we indicate the values that show the degree of expertise the user possesses in these concepts. To specify the degree of interest and expertise, we use set of linguistic variables: {*not interested, partially interested, interested*} and {*novice, medium, expert*}. We associate these variables with fuzzy sets, which are defined by the corresponding membership functions[16, 9].

The membership functions are based on the concept frequency value, which denotes importance of a certain concept for the user. To compute this value we need to know what concepts and how many times the user has encountered in the past. For this purpose, we analyze content of the pages accessed by the user and extract certain named entities using the Calais Web service and UIMA framework. These two analysis engines extract such entities as industry term, technology, company, location, etc. Occurrences of those entities are recorded in the user log, which allows us to estimate importance of a concept by computing the concept frequency value:

$$CF_{i,j} = \frac{c_{i,j}}{\sum_k c_{k,j}} \quad (1)$$

where c is the number of occurrences of the concept$_i$ for the user$_j$, and the denominator is the total number of occurrences of all concepts registered for the user$_j$. The computed concept frequency is a real number that can take values from 0 to 1. We use this value to define the membership functions that represent degree of user interest; μ_{ni}, μ_{pi}, μ_i show the degree to which the user is *not interested*, *partially interested*, and *interested* in the given concept (Figure 2). In the same way we define the membership functions to represent the degree of user expertise; μ_n, μ_m, μ_e show the degree of the user being *novice*, *medium*, and *expert* in a given concept (Figure 3).

We are currently investigating the possibilities to obtain the numbers showing the borders of the fuzzy sets. We

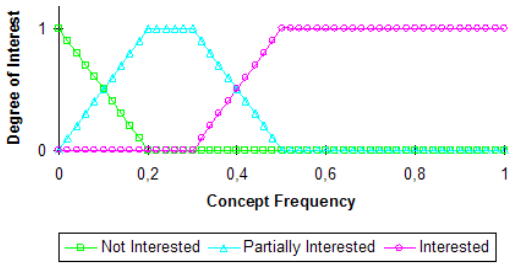

Figure 2. Membership functions for user interests

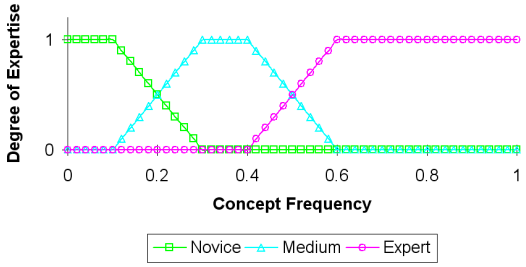

Figure 3. Membership functions for user expertise

plan to conduct a series of experiments to derive the correlations between the concept frequency values and the membership functions. This will allow us to correctly represent the degrees of user interests and expertise. Also we have discovered that computing concept frequency values for a group of similar concepts independently from the other groups will allow obtaining more accurate degrees of membership. E.g., for computing the concept frequency of *Finance* (meaning an industry sector), we should count only the occurrences of terms that denote industry sectors and neglect the other concepts; whereas for computing the concept frequency value of *Northern Europe* (meaning a location), we should count only the occurrences of the terms that denote locations.

In our user model, we capture not only the degrees of interest and expertise, but also the evolution of these two features over a certain period of time. The user log stores the time when the user encountered concepts. The time value is used to discover certain patters of changing user interest in a certain concept. For instance, this would allow revealing that at the end of every quarter and at the end of every year a manager of finance department is interested in obtaining the company's ROI report. Therefore, we can define a personalization rule that governs to deliver a mashup showing the ROI metrics to the manager at the appropriate time when she is interested in this information.

7. Personalization Model

In the *Personalization Model*, we define the personalization rules that govern what and how the mashup content is provided to the user. These rules basically define which information-gathering actions should be delivered to a user with certain interests and knowledge when she encounters a certain concept in the text she is reading. The personalization rules are specified in the Event-Condition-Actions [3] form as shown in Listing 2, where *event* denotes a situation when the user encounters a certain concept in the document she is reading, *condition* is combination of user features and context descriptions, and *actions* are the information gathering-actions that should be provided to the user when the event occurs.

```
on
        (event)
if
        (condition)
then
        (actions)
```

Listing 2: Personalization Rule Formula

In order to combine different user features and context descriptors, we represent them as dimensions. In its turn, the actions are specified at the intersections of these dimensions. For instance, in order to represent a personalization rule for an event when a student of a business school, interested in banking, with no knowledge in this field, is reading a news article that contains a bank in the text, we need to create three dimensions: *User Interests*, *User Expertise*, and *Document Concept* and plot values 'Banking', 'Novice', and 'Bank' respectively. At the intersection of these values, we specify what information should be delivered to the user, which in this case, could be the website of the bank, an encyclopedia article about the bank, and news related to this bank (Figure 4).

A multidimensional representation of the user and context features has two advantages. The first advantage is that we are enabled to use an arbitrary number of dimensions to define the personalization rules. In addition to the the interests, expertise, and the document concepts, we can consider such dimensions as the user location, device, accessibility characteristics, etc. Also we can add the time dimension to specify the personalization rules for a certain period of time. Consider a manager who at the beginning of every year travels for a regular corporate meeting to the headquarter of her company located in Stuttgart. Our user model defines that the user interest in Stuttgart is increasing in January. Now she is reading an email confirming the date on which the corporate meeting will take place this year. While reading the email, the manager might want to check availability of rooms in hotels of Stuttgart. To predict this

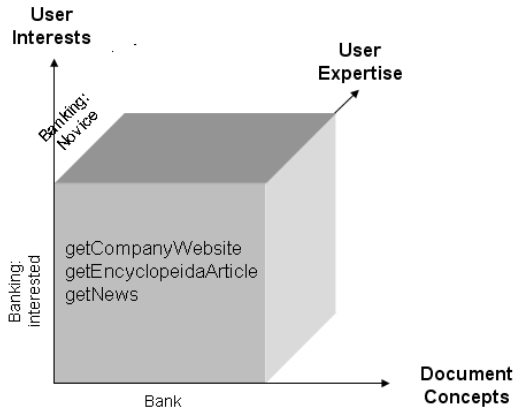

Figure 4. An example of multidimentional representation of personalization rules

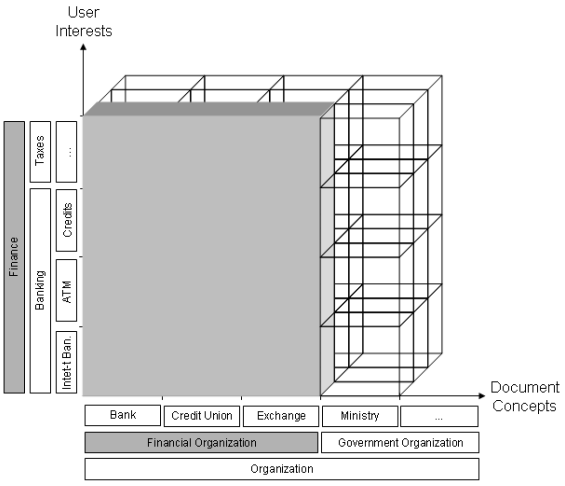

Figure 5. Personalization rules inheritance

information need and provide the user with the information she may need, we can generate and display a map mashup showing hotels that have vacant rooms on the date the meeting takes place. In order to generate such a mashup, we need to define a time-dependent personalization rule shown in Listing 3, which governs that when the user is reading a document that contains Stuttgart in the text and the current date is a day in January, then the framework should generate a mashup displaying a list of hotels in Stuttgart on the date mentioned in the text.

```
on
      (documentConcept = Stuttgart)
if
      ((interest = Stuttgart) AND
      (month = January))
then
      getHotels(location, checkinDate, checkoutDate)
```

Listing 3: Personalization Rule Example

Another advantage is the inheritance of the personalization rules. As described above, our domain model is represented as an ontology, where certain concepts are connected by the *isA* relationship. We harness this type of relationship in our personalization model to define personalization rules for top-level concepts, which could be inherited by their descendants. For example, we can specify a personalization rule for the document concept *FinancialOrganization* that governs to deliver two actions: *getAddress* and *getWebsite*. In the domain ontology, *FinancialOrganization* is defined as a super concept for *Bank*, *CreditUnion*, and *Exchange* (Figure 5). Therefore, we do not need to explicitly specify personalization rules for the subconcepts because they are automatically inherited from their super concept unless we need more specialized rules for one of the subconcepts.

The inheritance could be achieved at any of the dimensions that define their concepts in a taxonomy like structure. As we define the user interests and expertise using the domain ontology, we are able to achieve the inheritance of personalization rules at these dimensions too.

8. Conclusion

Augmenting the information provided to a portal user by the seamless integration of additional internal or external information sources via mashups seems to be a promising approach to fulfill users' information needs. In order to ensure that the provisioning of this additional information is tailored to the user and her current situation, one cannot rely on preexisting mashups. In order to support non-technical users, one can also not rely on the user to create these mashups herself. Therefore, in this paper, we propose an approach to automatic mashup generation based on multidimensional personalization rules that exploit information gathered in ontology-based domain, user, and task models. We are currently in the process of prototypically embedding this approach into IBM's WebSphere Portal.

References

[1] P. Achananuparp, H. Han, O. Nasraoui, and R. Johnson. Semantically enhanced user modeling. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 1335–1339, New York, NY, USA, 2007. ACM.

[2] P. Brusilovsky and E. Millän. User models for adaptive hypermedia and adaptive educational systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, chapter 1, pages 3–53. Springer, Berlin, Heidelberg, 2007.

[3] C. A.-N. Consortium. The active database management system manifesto: a rulebase of adbms features. *SIGMOD Rec.*, 25(3):40–49, 1996.

[4] B. Crabtree and S. J. Soltysiak. Identifying and tracking changing interests. *International Journal on Digital Libraries*, 2(1):38–53, October.

[5] O. Díaz, S. Pérez, and I. Paz. Providing personalized mashups within the context of existing web applications. In *WISE*, pages 493–502, 2007.

[6] R. Ennals, E. Brewer, M. Garofalakis, M. Shadle, and P. Gandhi. Intel mash maker: join the web. *SIGMOD Rec.*, 36(4):27–33, 2007.

[7] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Advanced Information and Knowlege Processing. Springer, 2003.

[8] R. I. John and G. J. Mooney. Fuzzy user modeling for information retrieval on the world wide web. *Knowl. Inf. Syst.*, 3(1):81–95, 2001.

[9] A. Kavcic. Fuzzy user modeling for adaptation in educational hypermedia. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(4):439–449, Nov. 2004.

[10] G. B. Lamont and D. A. V. Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[11] R. Lerner. At the forge: Creating mashups. *Linux J.*, 2006(147):10, 2006.

[12] C. D. C. Pereira and A. Tettamanzi. An evolutionary approach to ontology-based user model acquisition. In V. D. Gesu‘, F. Masulli, and A. Petrosino, editors, *WILF*, volume 2955 of *Lecture Notes in Computer Science*, pages 25–32. Springer, 2003.

[13] A. Schmidt. Enabling learning on demand in semantic work environments: The learning in process approach. In J. Rech, B. Decker, and E. Ras, editors, *Emerging Technologies for Semantic Work Environments: Techniques, Methods, and Applications*. IGI Publishing, 2008.

[14] D. E. Simmen, M. Altinel, V. Markl, S. Padmanabhan, and A. Singh. Damia: data mashups for intranet applications. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1171–1182, New York, NY, USA, 2008. ACM.

[15] M. D. Thang, V. Dimitrova, and K. Djemame. Personalised mashups: Opportunities and challenges for user modelling. In C. Conati, K. F. McCoy, and G. Paliouras, editors, *User Modeling*, volume 4511 of *Lecture Notes in Computer Science*, pages 415–419. Springer, 2007.

[16] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.

[17] H. Zhang, Y. Song, and H.-T. Song. Construction of ontology-based user model for web personalization. In *User Modeling 2007*, pages 67–76, 2007.