Undefined 0 (0) 1
IOS Press

Multi-application Profile Updates Propagation: a Semantic Layer to improve Mapping between Applications

Nadia Bennani [a], Max Chevalier [b], Elöd Egyed-Zsigmond [a], Gilles Hubert [b] and Marco Viviani [a]
[a] *Université de Lyon – LIRIS UMR 5205 – INSA de Lyon*
7, Avenue Jean Capelle – Bât Blaise Pascal – 69622 Villeurbanne Cedex, France
E-mail: {nadia.bennani, elod.egyed-zsigmond, marco.viviani}@insa-lyon.fr
[b] *Université Paul Sabatier – IRIT/SIG*
118, route de Narbonne – F-31062 Toulouse cedex 09, France
E-mail: {max.chevalier, gilles.hubert}@irit.fr

Abstract. In the field of multi-application personalization, several techniques have been proposed to support user modeling. None of them have sufficiently investigated the opportunity for a multi-application profile to evolve over time in order to avoid data inconsistency and the subsequent loss of income for web-site users and companies. In this paper, we propose a model addressing this issue and we focus in particular on management of user profile data propagation, as a way to reduce the amount of inconsistent user profile information over several applications. A second goal of this paper is to illustrate, in this context, the benefit obtained by the integration of a Semantic Layer that can help application designers to automatically identify potential attribute mappings between various applications. This paper so illustrates a work-in-progress work where two different approaches are integrated to improve a main goal: managing multi-application user profiles in a semi-automatic manner.

Keywords: User Profile, Data Propagation Management, User Profile Evolution, Multi-Application Personalization, Mappings identification, Semantic Layer, Log Analysis, Clustering

1. Introduction

Nowadays, many applications in different areas (digital libraries, search engines, e-learning, online databases, e-commerce, social networks...) are concentrating on collecting information about users for service personalization. For this reason, different applications in different areas (or within the same area) organize user properties, preferences and assumptions based on the user state, in *user profiles*. Each application manages user information independently from others, using a specific *user model*.

Information collection can be (i) *explicit*: information can be gathered by direct intervention of users themselves by filling some kind of predefined forms and/or (ii) *implicit*: information can be derived by studying users' behavior while using services (tracing).

When user profile management takes place in an isolated way at single-application level, we are in presence of *mono-application* scenarios. In such a case, *data incoherence* among isolated user profiles can be produced, due to several drawbacks strictly connected to mono-application personalization as *redundancy*: (i) it means a redundant process for users to re-enter their information every time they start using a new application and (ii) the same data for the same user are repeated several times and fragmented over many different applications, leading to data redundancy and tedious update; *lack of efficacy*: data connected to a given user remain private to each application. Even if a sufficient amount of data (or useful data) for the user has

been already collected by other applications, the user will not take advantage of it in the application he is currently using; *lack of experience*: as the user cannot take advantage of his information scattered across different applications, in the same way he cannot profit of the experience already accumulated by other users, in the same or different applications; *lack of control*: users have little or no control over the information defining their profiles, in particular over personalization and sharing, since their data are deeply buried in personalization engines. No accessibility protocols are given to users in order to manage their data.

In this paper we address these issues, discussing the possibility and the manner for user profile information to *evolve* in a multi-application context by user data *propagation*. We describe G-Profile, our *multi-application user modeling system* [29] and we also illustrate the possibility to integrate a novel feature in order to help application designers to identify user profile mappings between applications. This aspect is very important in G-Profile, where the condition for an efficient data propagation is a correct mapping generation phase. For this reason, to improve mapping management and to limit human intervention, we propose to add to G-Profile a *Semantic Layer*: a module allowing to automatically identify these mappings.

To illustrate our technique, we refer essentially to situations of *collaborating* applications, establishing *data relationships* between them without taking into account users' explicit choices and data propagation among different users. We will detail in the rest of the paper how the concepts of "collaboration" and "relationships among data" are really expressed. A concrete example to illustrate the utility of our multi-application user modeling system can be based on the scenario of virtual marketplaces, strictly connected to users' centers of interest. Let us imagine that a web search engine lands a deal with a virtual marketplace in order to share part of a user web search history with it. This way, the e-commerce application can exploit that, for instance, the user has been searching for specific items, in order to adjust its recommendation algorithm and present items regarding the centers of interest of the user at a given moment. Via our multi-application user modeling, it would be possible for users and virtual marketplaces to take advantage of the suggestion of new items according to up-to-date users' interests during the time.

The rest of the paper is organized as follows: in Section 2 we provide a short survey on current research in the field of personalization in multi-application environments. In Section 3 we describe G-Profile and the formalization of our model. We explain the concept of application collaboration via G-Profile and the way user data are propagated. Section 4 illustrates the Semantic Layer that can be integrated into G-Profile to improve the automatic identification of potential mappings between applications. Then, Section 5 discusses the benefits of the integration between G-Profile and the Semantic Layer. Finally, in Section 6 we present the main directions for future research and the conclusions.

2. Background

Efforts in *mono-application personalization* date back to the end of 1970's. Kobsa in his survey on Generic User Modeling Systems (GUMS) [20] describes several approaches applied in academic and commercial applications until the beginning of the 2000's.

The problem of personalization is addressed today in a scenario where distributed software environments are no longer static stand-alone applications, but dynamic integrative environments that configure themselves according to the individual needs of the user, the context of use, and the platform requirements. *User modeling* [20, 27] plays a crucial role in this kind of scenario and represents the basis for *multi-application* (cross-system) *personalization* [23].

In order to comprehensively integrate user information across different systems, one of the main challenges for user modeling is (represented by) guaranteeing *interoperability* of personalization approaches [5, 9]. From the literature, we outline two major approaches for user modeling interoperability in a multi-application scenario: (i) *standardization-based user modeling*, based on a *top-down* vision, defining some a priori – often centralized – *standard* whom all the involved applications have to comply; (ii) *mediation-based*[1] *user modeling*, behaving in a *bottom-up* way, operating a sort of *reconciliation* between different user model representations. It deals with transferring user modeling data from one representation to another, in the same domain, or across domains.

[1]The term 'mediation' has been introduced by Berkovsky *et al.* in [4]

2.1. Related Work

Standardization-based user modeling techniques are based on the definition of standard ontologies [10, 13, 18, 19, 25] and/or *unified* (general) user models [14, 15, 23, 24] which can be used with multiple systems. Standardization-based user modeling is therefore focused on the *reusability* of the user model itself.

Over the years, due to the great deal of syntactical and structural differences between existing user modeling systems, it has become clear that developing a commonly accepted full ontology of a domain, or envisaging all possible purposes for user modeling in all possible contexts, do not represent feasible solutions for multi-application personalization. A possible solution consists therefore in using mediation-based techniques, *mapping* different user model representations by the use of suitable *mapping rules* and/or *metamodels*. *Berkovsky et al.* in [4] give a formal definition for *mediation* of user models as "a process of importing the user modeling data collected by other (remote) [...] systems, integrating them and generating an integrated user model for a specific goal within a specific context". First attempts in this direction were done in [2, 6, 17].

Specific demonstrations of multi-agent based user modeling systems are given in [7, 16, 21, 22, 26, 28]. In [3] authors suggest the integration between mediation-based techniques and standardization of user modeling based on Semantic Web technologies (WordNet [30], GUMO and UserML). The use of semantics for integration of user models is proposed also in [8, 9].

All the techniques described before, address in different ways how to integrate a large number of available user model fragments for personalized service delivery, but fail in addressing the problem of user profile evolution, not considering the concept of user data propagation among user profiles over the time.

3. G-Profile

The aim of G-Profile is to provide a general-purpose and flexible user modeling system for multi-application environments. With respect to techniques already proposed in the literature, our approach is intended to address (i) *user profile evolution* via *user data propagation* in a (ii) *secure* and *user-centered* way.

As introduced before, in this paper we are interested in particular in describing the G-Profile role in guaranteeing correct propagation of user profile data modifications, disregarding issues on security, privacy, user behavior, and user preferences as well.

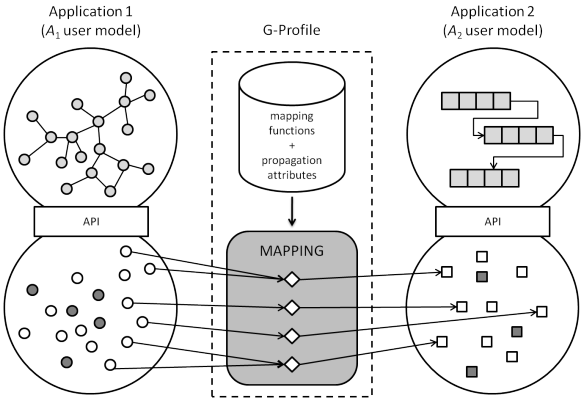

Fig. 1. Accessibility-based mapping between applications.

G-Profile does not propose neither a specific reconciliation technique able to take into account all the possible user data representations in different applications, nor a standard user profile model. Instead, it is an abstract and flexible protocol (Figure 1) able to interact with the potentially adopted matching techniques.

To do this, we define some abstract *mapping functions*, based on the generic concept of *mapping* between user data among applications[2].

Therefore, an application is *G-Profile-aware* if it provides a suitable *application programming interface* (API) to access both its user profile attributes and a set of mapping functions for these attributes to be used in mapping generation assisted by G-Profile.

3.1. User Profile Formalization

Each application A manages a set of user attributes a_k^A ($k \in \{1, \ldots, m_A\}$, where m_A is the total number of attributes for the application A). We assume that, for each user u_x using the application A, each attribute a_k^A has a value v_k associated, forming the *user profile element* as a couple (*attribute*, *value*). Formally, $e_k^{A,u_x} = \langle a_k^A, v_k \rangle$.

3.2. Data Mapping Formalization

In our system, where we plan to link attributes belonging to different applications, each attribute can,

[2]Once the generic mapping established via our protocol, it is possible to use semi-automatic assistants for the generation of concrete mappings, able to act either on structure-based or content-based collaborative user profiles. This aspect is out of the scope of this paper.

from time to time, be involved as the *source* or the *target* attribute in a relation with others. More specifically, since attributes are organized differently in each application A_i depending on the adopted user model, they can be permuted in several *source sets* $S_l^{A_i}$, $1 \leq l \leq 2^{m_{A_i}}$. Formally $S_l^{A_i} = \left\{s_1^{A_i}, s_2^{A_i}, \ldots, s_{t_{A_i}}^{A_i}\right\}$, where t_{A_i} is the total number of *source attributes* for the set $S_l^{A_i}$ belonging to the application A_i.

In the same way, each attribute of the application A_i can be a *target attribute* belonging to the target set T^{A_i} of the application A_i, such that $T^{A_i} = \left\{t_1^{A_i}, t_2^{A_i}, \ldots, t_{v_{A_i}}^{A_i}\right\}$ where v_{A_i} is the total number of target attributes for the set T^{A_i} belonging to the application A_i.

We define a *mapping* $\mathscr{M}^{A_i,A_j}$ *between two applications* A_i and A_j, $i \neq j$, as the triple $\mathscr{M}^{A_i,A_j} = \langle \mathbf{S}^{A_i}, T^{A_j}, M^{A_i,A_j} \rangle$. $\mathbf{S}^{A_i}$ is the set of source sets $S_l^{A_i}$, M^{A_i,A_j} is the set of *mapping functions* $m_k^{A_i,A_j}$ *between the applications* A_i and A_j associating to each source set $S_l^{A_i}$ a target attribute in T^{A_j}. Formally, $m_k^{A_i,A_j} : S_l^{A_i} \rightarrow t_h^{A_j}$.

We assume that the number k of mapping functions is *equal* to the number of target attributes of T^{A_j}.

3.3. Mapping Graph Formalization

It is possible to define a *graph* $\mathscr{G}$ as a *combination* of all the mappings in our environment. More specifically, we define our graph as a pair $\mathscr{G} = (V, E)$ composed of (i) a set V of *nodes*, (ii) a set E of *directed edges*. Our graph is therefore a *directed graph*.

We define two kinds of node: *attribute nodes* (`n-att`) and *function nodes* (`n-fun`). Formally $V = V_{\texttt{n-att}} \cup V_{\texttt{n-fun}}$.

In particular, we represent the elements of each $S_l^{A_i}$ and $t_h^{A_j}$ as `n-att` nodes, while the elements $m_k^{A_i,A_j}$ are represented as `n-fun` nodes. Formally, $S_l^{A_i} \in V_{\texttt{n-att}}$, $t_h^{A_j} \in V_{\texttt{n-att}}$, $m_k^{A_i,A_j} \in V_{\texttt{n-fun}}$.

We also define a function $\texttt{nodeType} : V \rightarrow \{\texttt{n-att}, \texttt{n-fun}\}$ that retrieves the type of a given node.

In the same way we define a function $\texttt{application} : V_{\texttt{n-att}} \rightarrow \mathbf{A}$ retrieving, for a given attribute node, the application it belongs to.

We represent an *edge* between two nodes n_1 and n_2 as $(n_1, n_2) \in E$, such that:

- $\forall (n_1, n_2) \in E \Rightarrow \texttt{nodeType}(n_1) \neq \texttt{nodeType}(n_2)$,
- if $(n_1, n_2) \in E$, $(n_2, n_3) \in E$ and $\texttt{nodeType}(n_2) = \texttt{n-fun} \Rightarrow \texttt{application}(n_1) \neq \texttt{application}(n_3)$.

In order to construct this graph, we define the algorithm 1.

Algorithm 1 Construction of the graph $\mathscr{G}$ from the set of existing mappings

Input: All the mappings $\mathscr{M}^{A_i,A_j}$
Output: $\mathscr{G}$

```
1: G = ∅
2: for all M^{Ai,Aj} do
3:     for all m_k^{Ai,Aj} ∈ M^{Ai,Aj} do
4:         add m_k^{Ai,Aj} to V
5:         for all s_g^{Ai} ∈ S_l^{Ai} do
6:             if s_g^{Ai} ∉ V then
7:                 add s_g^{Ai} to V
8:             end if
9:             add (s_g^{Ai}, m_k^{Ai,Aj}) to E
10:        end for
11:        if t_h^{Aj} ∉ V then
12:            add t_h^{Aj} to V
13:        end if
14:        add (m_k^{Ai,Aj}, t_h^{Aj}) to E
15:    end for
16: end for
```

3.4. Active Mapping

Let us consider two applications A_i and A_j, $i \neq j$, connected via a mapping $\mathscr{M}^{A_i,A_j}$. Let us suppose that a modification occurs on a source object $s_g^{A_i} \in S_l^{A_i}$, where $S_l^{A_i}$ is connected via a mapping function $m_k^{A_i,A_j}$ to an element $t_h^{A_j}$. It is our idea that the modification on $s_g^{A_i}$ will be propagated – via G-Profile – to $t_h^{A_j}$ *only if* some *conditions* imposed by the application A_j hold. To do this, every time a modification takes place on $s_g^{A_i}$, a set

$$(\mathrm{A})^{s_g^{A_i}} = \left\{(\alpha)^o, (\alpha)^t, (\alpha_1)^{s_g^{A_i}}, (\alpha_2)^{s_g^{A_i}}, \ldots, (\alpha_{n_{s_g}})^{s_g^{A_i}}\right\}$$

of *propagation attributes*, connected to $s_g^{A_i}$, is transmitted to G-Profile. This set contains *always* the identification of the application A_i at the origin of the modification. This information is detained by the attribute denoted as $(\alpha)^o$, that we will call *origin of the modification*. In the same way, the set always contains the *absolute modification time* attribute, denoted as $(\alpha)^t$.

It represents the instant (in absolute terms) wherein the original modification occurs.

Each application A_j defines, for each of its target elements $t_h^{A_j}$, a set $(\mathrm{K})^{t_h^{A_j}} = \left\{(\kappa_1)^{t_h^{A_j}}, (\kappa_2)^{t_h^{A_j}}, \ldots, (\kappa_{n_{t_h}})^{t_h^{A_j}}\right\}$ of *propagation conditions*.

Each propagation condition $(\kappa_{i_{t_h}})^{t_h^{A_j}}$ is a *boolean predicate* which can be based on the set $(\mathrm{A})^{s_g^{A_i}}$ or directly on A_j's rules.

This way, we can define a boolean function that we will call *mapping activation function* f acting on $(\kappa_1)^{t_h^{A_j}}, (\kappa_2)^{t_h^{A_j}}, \ldots, (\kappa_{n_{t_h}})^{t_h^{A_j}}$. The propagation of a change on $s_g^{A_i}$ to $t_h^{A_j}$ using $m_k^{A_i,A_j}$ is enabled if $f\left((\kappa_1)^{t_h^{A_j}}, (\kappa_2)^{t_h^{A_j}}, \ldots, (\kappa_{n_{t_h}})^{t_h^{A_j}}\right)$ is *true*.

Procedure

1. A modification occurs on $s_g^{A_i} \in S_l^{A_i}$;
2. G-Profile is notified that $s_g^{A_i}$ has been modified and it gets the new value associated to $s_g^{A_i}$, together with the propagation attributes;
3. G-Profile verifies the existence of a mapping function on $t_h^{A_j}$ having $s_g^{A_i}$ as source object;
4. G-Profile asks the application A_i for complementary data if the target object $t_h^{A_j}$ detains a matching function needing additional data;
5. Once all the needed source data are available, G-Profile sends to the application A_j: (i) the modification on $s_g^{A_i}$, (ii) possible additional data necessary to the mapping function involving $s_g^{A_i}$ and $t_h^{A_j}$, (iii) the list of propagation attributes given by the application A_i;
6. Once the application A_j receives this information from G-Profile, A_j will use them in order to evaluate the conditions that will effectively permit to propagate the modification on $s_g^{A_i}$ to $t_h^{A_j}$.

3.5. Recursive Active Mapping

As we have seen before, a modification can be propagated between *two* applications: from A_i to A_j if they are connected via a mapping $\mathscr{M}^{A_i,A_j}$. But the target element $t_h^{A_j} \in A_j$ can, in turn, be the source object $s_g^{A_j} \in S_l^{A_j}$ of a mapping function $m_k^{A_j,A_{j'}}$ $(j \neq j')$ connecting $S_l^{A_j}$ to $t_h^{A_{j'}} \in A_{j'}$, and so on. A modification occurred on A_i can, this way, propagate between several applications $A_j, A_{j'}, A_{j''} \ldots$ In this kind of scenario, two aspects need particular attention, notably (i) *cycles* and (ii) *parallelism*.

Concerning the first aspect, how to prevent *cyclical data propagation* in the presence for example of (i) symmetrical or (ii) cyclical mappings[3]?

Having established that $(\alpha)^o$ is a mandatory attribute to propagate with the modification itself, we automatically prevent this kind of situation. The origin of the modification does not change during the recursive propagation. It contains always the application that started the propagation, i.e., A_i. For this reason, between the propagation conditions, we introduce the mandatory presence of a condition $(\kappa)^o$ that checks the value of the $(\alpha)^o$ attribute. This way, we automatically stop the propagation when the target application has, as $(\alpha)^o$ value, the same $(\alpha)^o$ value detained by the source application ($(\alpha)^o = A_i$ in the case of our example).

Concerning the second aspect, problems are connected to a correct *modification ordering*. How to prevent, for example, that two modifications originating from two different source objects occur simultaneously on the same target object? For each modification, we define a separate *propagation sequence* starting when the original modification takes place and stopping at the first target application for which $f\left((\kappa_1)^{t_h^{A_j}}, (\kappa_2)^{t_h^{A_j}}, \ldots, (\kappa_{n_{t_h}})^{t_h^{A_j}}\right)$ is *false*. This way, it is impossible to have two modifications on the same target object at the same time. In addition to this, let us consider the following scenario. Let us suppose that two applications A_i and A_k are mapped to an application A_j. Is it possible to prevent that a modification (acting on $s_{g'}^{A_i}$, propagating to $t_h^{A_j}$), occurred at the time $t+1$, be replaced at the time $t+\delta$ by a modification (acting on $s_{g''}^{A_k}$, propagating to $t_h^{A_j}$) occurred at time t? This might happen due to differences in path length between source and target applications (Figure 2). The introduction of the $(\alpha)^t$ attribute as mandatory in our propagation attribute set, permits to synchronize data propagation avoiding drawbacks connected to parallelism.

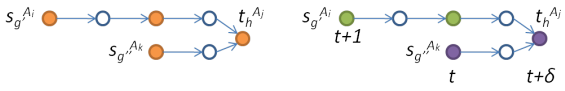

Fig. 2. Data propagation drawbacks connected to parallelism.

[3] e.g., (i) A_i is mapped to A_j and A_j is, in turn, mapped back to A_i; (ii) A_i is mapped to A_j, A_j is mapped to A_k and A_k is, in turn, mapped back to A_i or to A_j.

4. Semantic Layer to improve User Profile Mapping Identification

In the previous section, we have illustrated G-Profile and the motivations behind its development. When possible mappings have been identified between applications (i.e., between attributes constituting user profiles), possible modifications on user data connected to a specific user profile are propagated to other connected user profiles via some mappings. How these mappings are identified? One may propose that each application designer explains to others what corresponds to every shared attribute. This solution is quite limited because an application designer should give explanations to all applications with which she accepts to share user profiles. This may be a repeating and boring task when the number of applications grows. Moreover, sharing user profiles between applications is interesting only if these applications exploit the same kind of dimensions characterizing the user. As a solution, we propose to integrate in G-Profile a *Semantic Layer* allowing application designers to limit the manual detection of mappings between applications. This solution is based on previous works [11, 12] related to user profile interoperability between various applications. In the case of G-Profile, the Semantic Layer has the possibility to preliminarily know possible mappable attributes, this way its development is simplified with respect to the past.

4.1. Semantic Layer Content

Our Semantic Layer aims at describing the content of shared user profiles. This corresponds to a high-level description of various dimensions characterizing users in a specific context. For instance, such semantic description of user data can correspond to classical customer user profiles [1]. The customer profile includes attributes that can be classified in two main categories: *factual* attributes (i.e., Who is the customer?) and *transactional* attributes (i.e., What does the customer does?). An example of such description is illustrated in Figure 3. This description is more "conceptual" without considering how every user dimension is organized and implemented in each application. It corresponds to a dimension-based pivot format enabling the detection of attributes corresponding to the same concept even if they have different names, formats, ...

In addition to the conceptual description of user dimensions, the Semantic Layer includes the description of general *value-types* that are commonly used

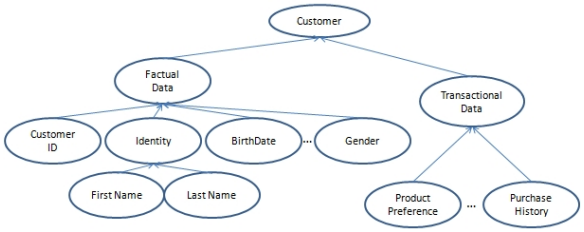

Fig. 3. Semantic Layer example characterizing customer dimensions.

in applications. One may consider some value-types like those used in XML Schema[4], highly sufficient for many applications where relations between value-types are expressed.

Furthermore, a set of conversion methods between types (i.e., 'yyyy-mm-dddd' => 'YYYY') is provided in order to improve the mapping identification process (i.e., restriction). This set could be adopted to provide higher interaction between the two layers, using these conversion methods for applications not having yet mapping functions.

The Semantic Layer can be defined in two ways:

1. Using an existing model framework available in various contexts;
2. Defining a high-level description of users, instead of defining a unique user profile structure for various applications. As we discussed at the beginning of the paper, this not represent nowadays a feasible solution due to heterogeneity of user models.

4.2. Using the Semantic Layer to Identify Potential Mappings

To allow the system to automatically identify potential mappings, every application have to select for each attribute it wants to share, the related concept and value-type information in the Semantic Layer (Figure 4). This figure shows only conceptual information, value-type information is not displayed for the sake of readability.

The Semantic Layer is used to identify related attributes via the application of a *transformation process*. This process has been adapted from [12], where multi-values are connected to attributes, because in G-Profile user profiles are supposed to be a list of attributes having a single value (see Section 3.1).

[4]http://www.w3.org/TR/xmlschema-2/

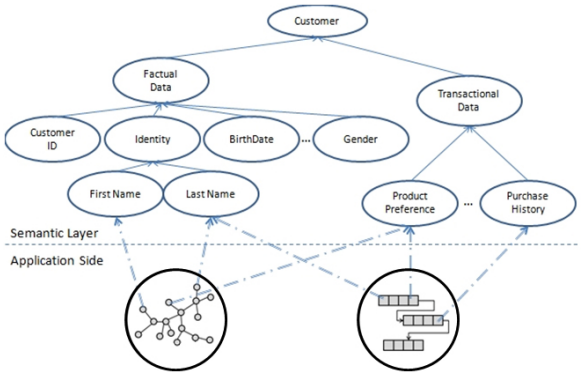

Fig. 4. User profiles associated to the Semantic Layer.

This process is illustrated in Figure 5. Four main steps can be identified:

1. Extraction of all the attributes shared by applications;
2. For every attribute, the associated conceptual information is also extracted from the Semantic Layer;
3. Thanks to these conceptual dimensions, for every couple of attributes the Semantic Layer is exploited to check if there is a possible relation between the concepts associated to these two attributes. To do this, inference rules are exploited to identify possible and accepted relations between concepts. All non-related attributes are deleted for further analysis.
4. For all the remaining couples of attributes, a value-type compatibility is verified. Thanks to information available in the Semantic Layer (value-type relationships and value-type conversion methods), the process verifies if the value-type of one attribute can be converted into the other one.

A potential mapping is automatically detected for every couple of attributes that passes all the first four steps.

4.3. Implementation

Identifying potential mappings between attributes coming from different applications can be based on an *ontology tool set*. Thus, the Semantic Layer can be based on a formal description through OWL[5]/RDF[6] to model relations between concepts and to store them. In order to verify the conceptual compatibility between two attributes (step (1.) described in the previous section) SparQL[7] is used.

[5]http://www.w3.org/TR/owl-ref/
[6]http://www.w3.org/RDF/

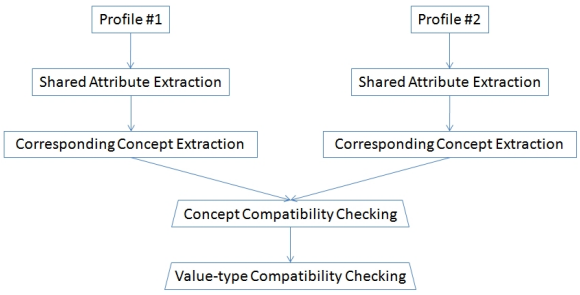

Fig. 5. Potential Mapping Identification.

```
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?c1  ?c2
FROM <Concepts.rdf>
WHERE
{
  ?c1 rdf:type sp:Concept .
  ?c2 rdf:type sp:Concept .
  { ?c1 owl:equivalentClass ?c2 }
   UNION
  { ?c1 owl:sameAs ?c2 } .
FILTER (
   (?c1=<cf1> || ?c1= <cf2>)
   &&
   (?c2=<cf1> || ?c2=<cf2>) )
}
```

Fig. 6. SparQL query example to verify conceptual compatibility.

Figure 6 illustrates a SparQL query example that verifies if two concepts `cf1` and `cf2` (associated to *attribute*1 and *attribute*2, respectively) are compatible through selected relationships between concepts (i.e., `equivalentClass` or `sameAs`). Note that the "." in the `WHERE` clause corresponds to the `AND` logical operator. Value-type compatibility checking relies on the same approach.

5. Benefits of Integrating G-Profile with the Semantic Layer

In this section, we illustrate via some use cases how the integration of the Semantic Layer can significantly improve the mapping generation/management in G-Profile. For the sake of simplicity, we consider three possible main actions in G-Profile: (i) an application

[7]http://www.w3.org/TR/rdf-sparql-query/

not using G-Profile wants to establish a partnership with a G-Profile-aware application; (ii) a user intends to be involved in a mapping generation/modification on user data she can effectively manage; (iii) two applications want to establish a partnership via G-Profile.

In the first scenario, a non-G-Profile-aware application that wants to establish a partnership with a G-Profile-aware one, must describe all the attributes it is effectively intended to share (i.e, every attribute has to be associated to the corresponding concept and its value-type in the Semantic Layer). The application then provides this description to G-Profile when establishing a partnership. The main advantage of such approach is that the description of the attributes is done only once during the attribute sharing decision phase.

The second case is more complex. As introduced before, a user can act only on a subset of user attributes that applications declare as public. On these data users can modify mappings already established by applications, or generate new mappings not provided by applications. In the first case, the semantic description of attributes can help users to manage/modify existing mappings. In the second case, not disposing of predefined mapping functions, the user is requested to play the role of 'application'. If she is intended to create a mapping between two applications where she has a user profile, she will memorize her data on a *personal data storage*, and how to map attributes connected to these data on the two different applications, could be done using the Semantic Layer described before.

Concerning the third action, the Semantic Layer is used to propose possible partnerships between applications. Thanks to the semantic description of the attributes provided by each application, G-Profile can, through the Semantic Layer, identify potential mappings between user profiles. On the basis of these potential mappings, mapping functions respecting semantic description can easily associate the attributes belonging to different applications thanks to G-Profile. Thus, building mapping functions can be considered as a simpler task with respect to the previous ones.

6. Conclusion and Perspectives

In recent years, several approaches have been proposed in different fields to solve the problem of multi-application personalization. The focus is gradually shifted from the model itself to the process of modeling. Due to the great deal of syntactical and structural differences between existing user modeling systems, developing a commonly accepted full ontology of a domain, or envisaging all possible purposes for user modeling in all possible contexts, definitely do not represent feasible solutions for multi-application personalization.

In this paper we have illustrated G-Profile: a flexible multi-application user modeling system, able to address typical problems of collaborative distributed environments and in particular to guarantee evolution, security, and privacy in multi-application personalization, aspects that have not been sufficiently considered up to now. We have focused in particular on the process of user profile data propagation in a multi-application environment. Data propagation relies for us on the correct identification of mappings between applications, mappings that are based on the generation of mapping functions between user attributes constituting user profiles. In order to facilitate both applications and users in the activity of generate and manage mapping functions, we have proposed to integrate to G-Profile a Semantic Layer allowing to identify potential mappings between applications based on a semantics. In addition to this, the Semantic Layer allows to better understand attributes definition in order to facilitate their use.

Our aim for the future is to complete our work by providing a complete evaluation, taking into account the user in the different phases of mapping establishment and data propagation, introducing the formalization and the methodology to address issues on security and privacy. Concerning aspects connected to the Semantic Layer, since the proposed approach allows a potential mapping identifications between user profiles even if they have different structures, G-Profile could benefit from such possibility by allowing applications to manage user profile through a more sophisticated structure (i.e., tree structure). Moreover, the current version of G-Profile relies on a single-value format for each attribute. Multi-valued attributes should be introduced in order to offer more possibilities to G-Profile-aware applications. Finally, a last perspective concerns the potential gain offered by the Semantic Layer. Indeed, a deeply reflection has to be done in order to implement generic transformation methods allowing G-Profile to directly map most of the attributes from one application to others. This could mainly simplify tasks related to mapping generation because most of them would be mainly generic.

References

[1] G. Adomavicius and A. Tuzhilin, "Using data mining methods to build customer profiles," *Computer*, vol. 34, pp. 74–82, February 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=619060.621654

[2] L. Ardissono, C. Barbero, A. Goy, and G. Petrone, "An agent architecture for personalized Web stores," in *Proc. of AGENTS '99*. New York, NY, USA: ACM, 1999, pp. 182–189.

[3] S. Berkovsky, D. Heckmann, and T. Kuflik, "Addressing challenges of ubiquitous user modeling: Between mediation and semantic integration," in *Advances in Ubiquitous User Modelling*, ser. LNCS. Springer, 2009, vol. 5830, pp. 1–19.

[4] S. Berkovsky, T. Kuflik, and F. Ricci, "Mediation of user models for enhanced personalization in recommender systems," *User Modeling and User-Adapted Interaction*, vol. 18, pp. 245–286, 2008, 10.1007/s11257-007-9042-9. [Online]. Available: http://dx.doi.org/10.1007/s11257-007-9042-9

[5] S. Berkovsky, T. Kuflik, and F. Ricci, "Cross-representation mediation of user models," *User Modeling and User-Adapted Interaction*, vol. 19, no. 1-2, pp. 35–63, 2009.

[6] D. Billsus and M. J. Pazzani, "User modeling for adaptive news access," *User Modeling and User-Adapted Interaction*, vol. 10, no. 2-3, pp. 147–180, 2000.

[7] A. Calì, D. Calvanese, S. Colucci, T. Di Noia, and F. M. Donini, "A description logic based approach for matching user profiles," in *Description Logics*, ser. CEUR Workshop Proc., vol. 104, 2004.

[8] F. Carmagnola, F. Cena, O. Cortassa, C. Gena, and A. Toso, "A preliminary step toward user model interoperability in the adaptive social web," in *Proc. of the UbiDeUm Workshop*, Corfu, Greece, 2007.

[9] F. Carmagnola and F. Cena, "User identification for cross-system personalisation," *Journal of Information Science*, vol. 179, no. 1-2, pp. 16–32, 2009.

[10] W. Chen and R. Mizoguchi, "Communication Content Ontology for Learner Model Agent in Multi-agent Architecture," in *Proc. of AIED-99 workshop on Ontologies for Intelligent Educational Systems*, 1999, pp. 95–102.

[11] M. Chevalier, C. Soulé-Dupuy, and P. L. Tchienehom, "Profiles semantics and matching flexibility for resources access," in *SITIS*, 2005, pp. 224–231.

[12] M. Chevalier, C. Julien, C. Soule-Dupuy, and N. Valles-Parlangeau, "Personalized information access through flexible and interoperable profiles," in *Web Information Systems Engineering Ű WISE 2007 Workshops*, ser. Lecture Notes in Computer Science, M. Weske, M.-S. Hacid, and C. Godart, Eds. Springer Berlin / Heidelberg, 2007, vol. 4832, pp. 374–385. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-77010-7_35

[13] P. Dolog and M. Schäfer, "A Framework for Browsing, Manipulating and Maintaining Interoperable Learner Profiles," in *User Modeling*, ser. LNCS. Springer, 2005, vol. 3538, pp. 397–401.

[14] European Telecommunication Standards Institute (ETSI), "Human Factors (HF) – User Profile Management," 2005, http://portal.etsi.org/STFs/STF_HomePages/STF342/eg_202325v010101p.pdf.

[15] Future of Identity in the Information Society (FIDIS), "D2.3 – Models," 2005, http://www.fidis.net/fileadmin/fidis/deliverables/fidis-wp2-del2.3.models.pdf.

[16] G. González, B. López, and J. Lluís de la Rosa, "A Multi-agent Smart User Model for Cross-domain Recommender Systems," in *Proc. of the IUI '05 Workshop on the Next Stage of Recommender Systems Research*, Edinburgh, UK, 2005.

[17] J. E. Greer, G. I. McCalla, J. Cooke, J. A. Collins, V. Kumar, A. Bishop, and J. Vassileva, "The Intelligent Helpdesk: Supporting Peer-Help in a University Course," in *Proc. of ITS '98*. London, UK: Springer, 1998, pp. 494–503.

[18] D. Heckmann, T. Schwartz, B. Brandherm, and A. Kroner, "Decentralized User Modeling with UserML and GUMO," in *Proc. of the UM '05 DASUM Workshop*, Edinburgh, UK, 2005.

[19] J. Kay, "Ontologies for reusable and scrutable student model," in *Proc. of AIED-99 workshop on Ontologies for Intelligent Educational Systems*, 1999, pp. 72–77.

[20] A. Kobsa, "Generic user modeling systems," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1-2, pp. 49–63, 2001.

[21] A. Lorenz, "A Specification for Agent-Based Distributed User Modelling in Ubiquitous Computing," in *Proc. of the UM '05 DASUM Workshop*, Edinburgh, UK, 2005.

[22] B. Mehta and W. Nejdl, "Intelligent Distributed User Modelling: from Semantics to Learning," in *Proc. of the UbiDeUm Workshop*, Edinburgh, UK, 2007.

[23] C. Niederée, A. Stewart, B. Mehta, and M. Hemmje, "A Multi-Dimensional, Unified User Model for Cross-System Personalization," in *Proc. of the AVI 2004 Workshop on Environments for Personalized Information Access*, Gallipoli, Italy, 2004.

[24] F. Petersen, G. Bartolomeo, M. Pluke, and T. Kovacikova, "An architectural framework for context sensitive personalization: standardization work at the ETSI," in *Proc. of Mobility '09*. New York, NY, USA: ACM, 2009, pp. 1–7.

[25] L. Razmerita, A. A. Angehrn, and A. Maedche, "Ontology-based user modeling for knowledge management systems," in *User Modeling*, ser. LNCS, vol. 2702. Springer, 2003, pp. 213–217.

[26] K. van der Sluijs and G.-J. Houben, "Towards a generic user model component," 2005, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.3148.

[27] J. Vassileva, "Distributed user modelling for universal information access," in *Proc. of the 9th Int. Conf. on Human-Computer Interaction)*, vol. 3. New Orleans, USA: Lawrence Erlbaum, 2001, pp. 122–126.

[28] J. Vassileva, G. Mccalla, and J. Greer, "Multi-agent multi-user modeling in i-help," *User Modeling and User-Adapted Interaction*, vol. 13, no. 1-2, pp. 179–210, 2003.

[29] M. Viviani, N. Bennani, and E. Egyed-Zsigmond, "Multi-application Personalization using G-Profile," *IJCSIS - Users and Information Systems, Special Issue*, 2011, to appear.

[30] Princeton University, "Wordnet – a lexical database for english," 2010, http://wordnet.princeton.edu.