IADIS International Journal on Computer Science and Information Systems
Vol. 6, No. 2, pp. 95-115
ISSN: 1646-3692

MULTI-APPLICATION PERSONALIZATION USING G-PROFILE

Marco Viviani, Nadia Bennani and Elöd Egyed-Zsigmond
Université de Lyon – LIRIS UMR 5205 – INSA de Lyon 7, Avenue Jean Capelle – Bât Blaise Pascal, 69622 Villeurbanne Cedex, France

ABSTRACT

Sharing heterogeneous data among distributed environments in a user-centric way represents today the main challenge for personalization. In recent years several techniques have been proposed to support user modeling for multi-application personalization. In this paper we describe G-Profile, our multi-application user modeling system. G-Profile represents a way to address some open issues not sufficiently investigated in literature, as the opportunity for a multi-application profile to evolve over the time, together with the possibility to guarantee security and privacy in the diffusion of user information among applications. In particular, this paper focuses on user profile data modifications propagation, aiming to reduce the amount of incoherent information about the user over several applications.

KEYWORDS

Multi-application personalization, user profile evolutivity, user data propagation, user-centricity, interoperability.

1. INTRODUCTION

Nowadays, many different applications in different areas (digital libraries, search engines, e-learning, online databases, e-commerce, social networks, etc.) 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, based on a specific *user model*.

Information collection can be (*i*) *explicit*: information is gathered by a direct intervention of the users themselves by filling some kind of predefined forms and/or (*ii*) *implicit*: information is 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 is produced, due to several drawbacks strictly connected to mono-application personalization:

1. *redundancy*: considering the number of different applications where each user can be involved, (*i*) it represents a redundant process to the user to re-enter her information every time she begins with 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;
2. *lack of efficacy*: due to the absence of collaboration between applications, data connected to a certain user remain private to each application. Even if a sufficient amount of data (or useful data) for the user herself has been already collected by other applications, the user will not take advantage of it in the application she is currently using;
3. *lack of experience*: as the user can not take advantage of her information scattered across different applications, in the same way she can not profit of the experience already accumulated by other users, in the same or different applications;
4. *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.

This paper introduces G-Profile: our *multi-application user modeling system* [38]. Our purpose is to give a solution to drawbacks connected to mono-application environments, specifically:

- the possibility and the manner for user profile information to *evolve* in a multi-application context by user data *propagation*;
- the possibility to control *security* and *privacy* by applications and users over personal data in the diffusion of user information among applications.

We describe, in particular, the way our model addresses the first item. We evaluate it by monitoring the *coherence* of user profile information over the applications managing her profiles.

1.1 Scenario

In this section we illustrate a concrete scenario where our multi-application user modeling system could be used to solve drawbacks connected to mono-application environments. Let us take as an example the domain of virtual marketplaces. Each of the following cases is directly referred to the corresponding drawback.

1. A user subscribes several marketplaces collaborating between them via the multi-application user modeling system. This way the user has the possibility to type her data as gender, date of birth, address, only once in the first marketplace she subscribes.
2. A user changes her address on her user profile on the marketplace *A*. The new zip code changes from *xxxx*1 to *xxxx*2. The same user has a profile on the marketplace *B*. On the marketplace *B*, the zip code for the user is still *xxxx*1. The marketplace *B* is currently promoting a discount for all the clients having *xxxx*2 as a zip code. Via the multi-application user modeling system, it would be possible for the user to take

advantage of the promotion as the new zip code value would have been propagated automatically.

3. A user α, having a profile on a marketplace *A*, has some friends $\beta, \gamma, \ldots, \omega$ using the same or different marketplaces collaborating between them via the multi-application user modeling system. If $\beta, \gamma, \ldots, \omega$ have favorite items, if they decide to share their favorite items with α, and if the marketplaces permit the sharing of this kind of data, the favorite items of α's friends can be accessible by α.
4. Users $\alpha, \beta, \gamma, \ldots, \omega$ are friends and use different marketplaces. These marketplaces collaborate between them via the multi-application user modeling system, but only for not-competing purposes. So, even if it could be interesting for users to share their favorite items, the different marketplaces can decide not to permit the sharing of certain kind of items that could advantage their competitors (the same items provided by a competitor for example). Even in the case of two collaborative marketplaces leaving open to users the possibility to share all their bought item list with friends, each user has always the possibility to restrict the access of this information to particular users. Moreover, the access to the bought item list of α can be restricted differently for β on various marketplaces where β is registered, based on the established collaboration between marketplaces and users.

Cases 1 and 2 refer essentially to situations of collaborating applications; in cases 3 and 4, also users' explicit choices are taken into account. In this paper, we describe how G-Profile operates in the first two cases.

Therefore, 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 our approach: we introduce the general idea and the formalization of our model. We explain the concept of "collaboration" among applications via G-Profile and the way user data are propagated. In Section 4 we evaluate the advantages of our technique connected to user data coherence. Finally, in Section 5 we present the main directions for future research and the conclusions.

2. BACKGROUND IN PERSONALIZATION

Efforts in mono-application personalization date back to the end of 1970's. Kobsa in his survey on Generic User Modeling Systems (GUMS) [24] 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* [24, 36] plays a crucial role in this kind of scenario and represents the basis for *multi-application* (cross-system) *personalization* [30].

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 [9, 6]. Applied to user modeling, interoperability can be seen as "the ability to access and interpret information derived from multiple heterogeneous sources and to integrate this information into a user model of proper granularity" [36, 9]. Two main

types of interoperability emerge from literature: (*i*) *syntactic interoperability*, as the ability for two or more systems of communicating and exchanging data, overtaking differences between information systems at the *application level* and (*ii*) *semantic interoperability*, it overcomes differences between information systems at the *knowledge level*, "creating a semantically compatible information environment based on the agreed concepts between different (…) entities" [31].

2.1 Related Work

From 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 user modeling* (the term 'mediation' has been introduced by Berkovsky *et al.* in [5]), 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.

2.1.1 Standardization-based User Modeling

Standardization-based user modeling techniques are based on the definition of standard ontologies and/or *unified* (general) user models which can be used with multiple systems. Standardization-based user modeling is therefore focused on the *reusability* of the user model itself.

Standard Ontologies First attempts to propose standard ontologies for user modeling were by Chen & Mizoguchi [11] and Kay [23] at the end of the 1990's. The first approach used ontologies for learner modeling and the second one motivated ontology-based reusable and understandable modeling of students. These approaches presented difficulties in the construction of the initial ontology and were applied to specific domains. The work of Razmerita *et al.* [34] has been the first to introduce the notion of generic ontology-based user model. Authors presented here OntobUM, a generic ontology-based user modeling architecture integrating three ontologies: one for the users, another one defining the relations

between the applications (the domain ontology) and the last one (the log ontology) defining the user-application interaction semantics. Dolog & Schäfer have proposed in [12] a framework providing a common base for the exchange of learner profiles between several sources, based on standards for learner modeling. A domain ontology and a learner ontology are used for sharing learner models. By the use of web service technologies, user models can be exchanged between different services by means of Java APIs. An extensive approach for ontology-based representation of user models has been proposed by Heckmann *et al.* by introducing GUMO [22], a General User Modeling Ontology for the uniform interpretation of distributed user models in intelligent Semantic Web enhanced environments. In order to address the problem of the uniform interpretation of decentralized user models, a new architecture employing UserML [20] (an XML-based user modeling mark-up language) and GUMO was presented in [21]. To date, GUMO seems to be the most comprehensive user modeling ontology proposed. Such ontology may be represented by modern Semantic Web languages, easing this way the user model exchange between different applications.

Unified User Models Concerning the construction of unified user models, Amato & Straccia have presented in [2] a quite abstract user profile model and discussed what

information should be represented in a user profile, with particular reference to digital libraries. Niederée *et al.* have introduced in [30] their Unified User Context Model (UUCM), a centralized and extensible multidimensional user model for aggregating the partial user models collected by individual personalization systems. The UUCM approach is based on a Context Passport, able to extract and integrate the required user information from user models via a Cross-System Communication Protocol (CSCP). Each personalization system has to build upon its user model the UUCM structure. In [29], the same authors have suggested the use of ontologies for the standardization of user models and for easing information exchange between applications. For existing personalization systems, it is necessary to share common vocabularies and ontologies and to support the CSCP protocol. In the effort of describing how the identity of a person can be represented in different domains and what are the processes used for dealing and manipulating it, the Future of Identity in the Information Society (FIDIS) has published a technical report [16] whose aim was to present different models of representation (categories of attributes or data schema) of a person across different application domains. To the same purpose, the European Telecommunication Standards Institute (ETSI) has published the guidelines on personalization and user profile management [13]. In this work the issue of how users can be provided with an integrated approach to their profiles was discussed. Over the years the same institution has worked on the development of a standardization-based architectural framework for user profile management. The approach proposed in [32] explains a solution for context sensitive automatic activation of user profiles, while still providing the user with the option to activate profiles manually.

2.1.2 Mediation-based User Modeling

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 cross-system personalization.

Mediation A possible solution consists therefore in using mediation-based techniques, *mapping* different user model representations by the use of suitable *mapping rules* and/or *meta-models*. Berkovsky *et al.* in [5] gives 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". It is the same author that in [4] better explains this definition: *integration* refers to the set of techniques aimed at resolving inconsistencies and heterogeneities among data; mediation enriches in a context-aware way existing user models in a given system by collected data obtained from remote systems. First attempts in this direction were done using multi-agent technology at the end of 1990's [18, 3, 7] with the introduction of *decentralized user modeling*, especially in the field of *ubiquitous computing* [19]. In a distributed multi-agent based software environment, the traditional centralized user model (centralized user modeling servers [14] were initially proposed to support adaptation in networked applications: the representation of the user model followed a particular centralized schema which was known in advance to the applications.) ceases to exist and it is replaced by user model fragments, developed by the various software agents populating the environment. The challenge in such a scenario is to go through *active user modeling* [27]: the integration of the large number of available inconsistent user model fragments for personalized service delivery. In such a scenario, it becomes fundamental for each approach to identify which are the suitable

matching/mapping techniques for the integration of the user model fragments in order to provide a sort of *on demand* user modeling.

Mediation Techniques A specific demonstration of a multi-agent based user modeling process is given in the work of Vassileva *et al.* [37]. In this approach, user modeling is viewed as a computing process over four dimensions: subjects, objects, purposes and resources. Even if it is not necessary to establish and maintain an isolated, centralized user modeling server, the proposed mapping technique (*matchmaking*) – where a variety of agents keep track of user models and map help requests to possible service providers – is based on a given domain taxonomy (or on a "catalogue of purposes for user modeling" as they stated in their later works). Another approach for matching user profiles has been proposed by Calì *et al.* in [8]. Authors describe an algorithm based on a description logic based language, for matching demands and supplies of profiles, taking into account incompleteness of profiles and incompatibility between demand and supply. The matching process is modeled as a special *reasoning* service about profiles. This approach is seemingly suitable for situations where user models operates in the same context. van der Sluijs & Houben in [35] present their Generic User model Component (GUC): different applications have to subscribe to a GUC in order to upload user data. This is possible via a *schema* describing the data structure of the user models for different applications. Data exchange between applications is guaranteed via some schema

mapping based on *data reconciliation rules*. Authors suggest the possibility to use different matching and merging techniques to map input schemas and create a merged schema as the union of the input schemas. They also propose to construct combined ontologies of the application schemas. SUM is a multi-agent Smart User Model proposed by Gonzalez *et al.* in [17]. Each application generates several context-based user models, one for each context. *Graph-based relationships* are used to map the context-dependent user models with the SUM. In the approach proposed by Lorenz in [26, 25] a flexible representation of a user is obtained by assembling the knowledge of all the agents reachable in the current context. The basic underlying cooperation-approach between agents is based on an *information-exchange* protocol. However, neither the sharing policy nor the conversion mechanisms between various user models is practically defined, so the aim of these works remain essentially to provide a well-defined conceptual basis. Mehta & Nejdl, in [28] present a methodology based on *machine learning techniques* for automatically mapping profiles. The use of the same techniques in addition to *data mining techniques* to enhance the mediation process is suggested by Berkovsky *et al.* in [5], while in [4] the same authors suggest the integration between mediation-based techniques and standardization of user modeling based on Semantic Web technologies (WordNet [33], GUMO and UserML). The use of semantics is proposed also in [10, 9]. In the first work is presented a *tag-based user model exchange approach* based

on the idea to give to different systems the possibility to exchange information connected to 'tag-enriched' user profiles. The second approach focuses more on the task of *identification* of the users exchanging data among systems, as a pre-condition for user knowledge exchange and user data integration.

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 literature, our approach is intended to address:

1. *user profifile evolutivity* via *user data propagation*;
2. *accessibility-based mapping* (on demand).

None of these two aspects have been sufficiently investigated, to our knowledge, in current research. Concerning the first aspect, even if some techniques address the reconciliation of multiple user data over the time, they do not take into account the possibility of automatic propagation of data changes among applications. The second aspect is even less considered: several ideas have been developed on the way mappings between data can be established, but none of them discuss the way to map data according to their degree of accessibility.

For these reasons, 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, we describe an abstract and flexible protocol able to interact with the potentially adopted matching techniques and to take into account the possibility to share data at a given level.

3.1 G-Profile-aware Applications

Figure 1 illustrates our idea: the G-Profile protocol does not impose any kind of constraints on the user model representation (in case) already defined by each application in a multi-application environment.

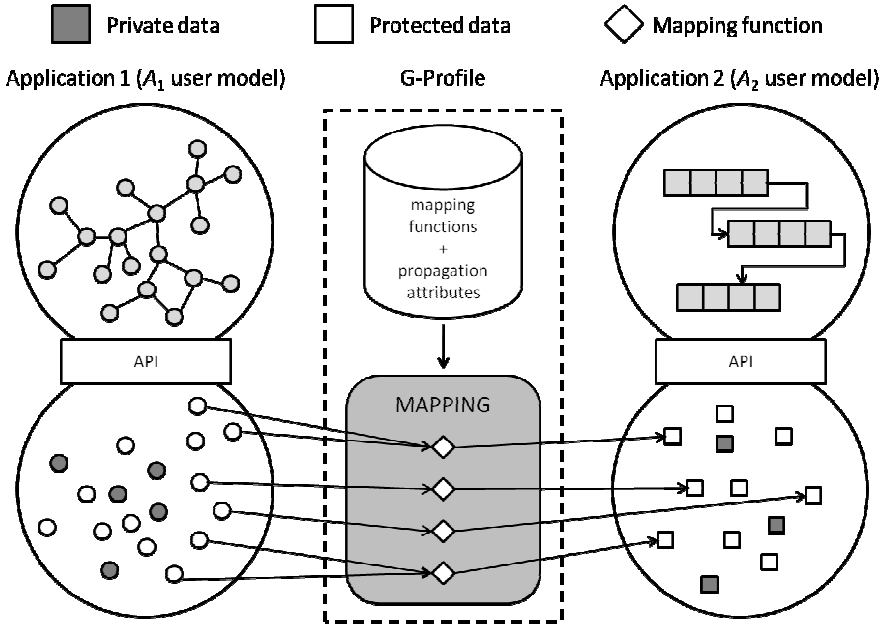

Figure 1. Accessibility-based mapping between applications.

In order to give to the user profile the possibility to reflect data changes happened in the whole system, we define some abstract *mapping functions*, based on the generic concept of *mapping* between user data among applications (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). We define the possibility to establish a mapping only between information that applications and users want effectively share among them.

To do this, each application is able to express its user model in conformity with a proposed *accessibility meta-classification* of the user profile information. This accessibility meta-classification defines two *accessibility degrees* given by each application: *private* data and *protected* data. Mappings at this level can be established only between protected data. At user level, this category can be managed by users themselves and yet again divided into *private* and *public* data according to the user preferences. Consequently, the data accessibility degree can change depending on the application and/or the other users the current user is dealing with.

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 accessibility-based mappings assisted by G-Profile.

G-Profile is responsible for the control of mapping establishment and user data propagation. It is in charge of data transmission and security and privacy issues. In this paper, we are interested in particular in its role in guaranteeing correct user profile data modifications propagation. For this reason, the formalization we provide in the following section concerns the evolutivity part of our technique, disregarding security and privacy aspects.

3.2 User Profile Formalization

Let $\mathbf{A}$ be a set of applications in a multi-application environment. Formally,

$$\mathbf{A} = \{A_1, A_2, \dots, A_n\}$$

Each application A_i ($i \in \{1, \dots, n\}$) manages a set D^{A_i} user attributes. Formally,

$$D^{A_i} = \left\{a_1^{A_i}, a_2^{A_i}, \dots, a_{m_{A_i}}^{A_i}\right\}$$

where m_{A_i} is the total number of attributes for the application A_i.

We assume that, for each user u_x using the application A_i, each attribute a_k $\left(k \in \{1, \dots, m_{A_i}\}\right)$ has a value v_k associated, forming the *user profile element* as a couple (*attribute*, *value*). Formally

$$e_k^{A_i,u_x} = \langle a_k^{A_i}, v_k \rangle \quad (1)$$

Consequently, we define the *user profile* $\boldsymbol{U}^{A_i,u_x}$ for each user u_x in the application A_i as

$$\boldsymbol{U}^{A_i,u_x} = \left\{e_k^{A_i,u_x}\right\}$$

for $k = 1, \dots, m_{A_i}$.

3.3 Data Mapping Formalization

According to [15], the *data mapping problem* consists in discovering effective mappings between structured representations of data. In general, given a set of *source objects* $S = \{s_1, s_2, \dots, s_t\}$ and a *target object* t, if the elements of S can be related through some *relation* to t ($S \to t$), we indicate this relation as m and we call it *mapping function*, acting on the couple $\langle S, t \rangle$. Formally,

$$S \xrightarrow{m} t \text{ or, simply } m: S \to t$$

Several source sets $S_1, S_2, \dots, S_r$ can be related with several target objects $t_1, t_2, \dots, t_v$ via different mapping functions $m_1, m_2, \dots, m_w$. Formally

$$\mathbf{S} = \{S_1, S_2, \dots, S_r\}, T = \{t_1, t_2, \dots, t_v\} \text{ and } M = \{m_1, m_2, \dots, m_w\}$$

Therefore, a *mapping* $\boldsymbol{M}$ is the triple

$$\boldsymbol{M} = \langle \mathbf{S}, T, M \rangle \quad (2)$$

3.3.1 Mapping between Applications

In our system, where data in each application A_i, $i \in \{1, \dots, n\}$, are organized differently depending on the adopted user model, the attributes of each application A_i can be permuted in several source sets $S_l^{A_i}$, $1 \le l \le 2^{|D_{A_i}|}$, each $S_l^{A_i}$ belonging to the source set $\mathbf{S}^{A_i}$. Formally

$$\mathbf{S}^{A_i} = \left\{ S_l^{A_i} | l: 1, \dots, 2^{|D_{A_i}|} \right\} \text{ and } S_l^{A_i} = \left\{ s_1^{A_i}, s_2^{A_i}, \dots, s_{t_{A_i}}^{A_i} \right\}$$

where t_{A_i} is the total number of source objects for the set $S_l^{A_i}$ belonging to the application A_i, $s_1^{A_i}, s_2^{A_i}, \dots, s_{t_{A_i}}^{A_i} \in D^{A_i}$ and $S_l^{A_i} \subseteq D^{A_i}$.

In the same way, each attribute of the application A_i can be a target object belonging to the target set T^{A_i} of the application A_i, such that

$$T^{A_i} = \left\{ t_1^{A_i}, t_2^{A_i}, \dots, t_{v_{A_i}}^{A_i} \right\}$$

where v_{A_i} is the total number of target objects for the set T^{A_i} belonging to the application A_i, $t_1^{A_i}, t_2^{A_i}, \dots, t_{v_{A_i}}^{A_i} \in D^{A_i}$ and $T^{A_i} \subseteq D^{A_i}$.

From (2) we define a *mapping* $\boldsymbol{M}^{A_i,A_j}$ *between two applications* A_i and A_j, $i \in \{1, \dots, n\}$, $j \in \{1, \dots, n\}$, $i \neq j$, as the triple

$$\boldsymbol{M}^{A_i,A_j} = \langle \mathbf{S}^{A_i}, T^{A_j}, M^{A_i,A_j} \rangle \quad (3)$$

where

$$M^{A_i,A_j} = \left\{ m_k^{A_i,A_j} \right\}$$

and $m_k^{A_i,A_j}$ is a *mapping function between two applications* associating to each source set $S_l^{A_i}$ a target element in T^{A_j}. Formally

$$m_k^{A_i,A_j}: S_l^{A_i} \to t_h^{A_j} \quad (4)$$

We notice that the number k of mapping functions is *equal* to the number of target objects of T^{A_j}, i.e., $k = 1, \dots, \left|T^{A_j}\right|$. Formally

$$\left|M^{A_i,A_j}\right| = \left|T^{A_j}\right|$$

3.3.2 Building a Mapping Graph

Let us indicate with **M** the set of all the mappings $\boldsymbol{M}^{A_i,A_j}$, $i \in \{1, \dots, n\}$, $j \in \{1, \dots, n\}$, $i \neq j$. Formally

$$\mathbf{M} = \left\{\boldsymbol{M}^{A_i,A_j} | i \in \{1, \dots, n\}, j \in \{1, \dots, n\}, i \neq j\right\}$$

It is possible to define a graph $\boldsymbol{G}$ as a combination $\boldsymbol{C}$ of all the mappings in our environment. Formally

$$\boldsymbol{G} = \boldsymbol{C}(\mathbf{M})$$

The combination $\boldsymbol{C}$ corresponds to the Algorithm 1 described later. More specifically, we define our graph as a pair $\boldsymbol{G} = (V, E)$ 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 possible kinds of node: *attribute nodes*: n_att and *function nodes*: n_fun. Formally

$$V = V_{n\_att} \cup V_{n\_fun}$$

In particular, we represent the elements of $S_l^{A_i}$ 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

$$S_l^{A_i} \in V_{n\_att}, t_h^{A_j} \in V_{n\_att}, m_k^{A_i,A_j} \in V_{n\_fun}$$

We also define a function $nodeType: V \rightarrow \{n\_att, n\_fun\}$ that retrieves the type of a certain node.

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

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

Remarks:

- $\forall (n_1, n_2) \in E \Longrightarrow nodeType(n_1) \neq nodeType(n_2)$
- if $(n_1, n_2) \in E$, $(n_2, n_3) \in E$ and $nodeType(n_2) = n\_fun \Longrightarrow application(n_1) \neq application(n_3)$

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

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

Input: All the mappings $\boldsymbol{M}^{A_i,A_j} \in \mathbf{M}$
Output: $\boldsymbol{G}$
1: $\boldsymbol{G} = \emptyset$
2: **for all** $\boldsymbol{M}^{A_i,A_j} \in \mathbf{M}$ **do**
3: **for all** $m_k^{A_i,A_j} \in \boldsymbol{M}^{A_i,A_j}$ **do**
4: add $m_k^{A_i,A_j}$ to V
5: **for all** $s_g^{A_i} \in S_l^{A_i}$ **do**

6: **if** $s_g^{A_i} \notin V$ **then**
7: add $s_g^{A_i}$ to V
8: **end if**
9: add $\left(s_g^{A_i}, m_k^{A_i,A_j}\right)$ to E
10: **end for**
11: **if** $t_h^{A_j} \notin V$ **then**
12: add $t_h^{A_j}$ to V
13: **end if**
14: add $\left(m_k^{A_i,A_j}, t_h^{A_j}\right)$ to E
15: **end for**
16: **end for**

3.3.3 Active Mapping

Let us consider two applications A_i and A_j, $i \neq j$, connected via a mapping $\boldsymbol{M}^{A_i,A_j}$. Let us suppose that a modification **m** occurs on the value v_k of an element $e_k^{A_i,u_x} = \langle a_k^{A_i}, v_k \rangle$. In this scenario, the element $e_k^{A_i,u_x}$ is represented by the 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 $t_h^{A_j}$. For this reason, for abuse of terminology, in the rest of this section we will say that a modification occurs on $s_g^{A_i}$.

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* certain *conditions* imposed by the application A_j hold. To do this, every time a modification takes place on $s_g^{A_i}$, a set

$$(\mathrm{A})^{s_g^{A_i}} = \left\{(\alpha)^o, (\alpha)^t, (\alpha)_1^{s_g^{A_i}}, (\alpha)_2^{s_g^{A_i}}, \dots, (\alpha)_{n_{s_g}}^{s_g^{A_i}}\right\}$$

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

$$(\mathrm{K})^{t_h^{A_j}} = \left\{(\kappa)_1^{t_h^{A_j}}, (\kappa)_2^{t_h^{A_j}}, \dots, (\kappa)_{n_{t_h}}^{t_h^{A_j}}\right\}$$

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 protocols.

Example 1. A modification takes place from the application A_1 on a source object $s_1^{A_1} \in S_1^{A_1}$. The 'origin of the modification' attribute $(\alpha)^o = A_1$ is passed to G-Profile. The target application A_2 can decide to evaluate two conditions in order to accept the propagation of the

modification. The first condition $(\kappa)_1^{t_1^{A_2}}$ is a boolean predicate based on the attribute $(\alpha)^o$, i.e., $(\kappa)_1^{t_1^{A_2}}$ = *distance in* $\boldsymbol{G}$ *from* $(\alpha)^o \leq$ *2 nodes*. The second condition $(\kappa)_2^{t_1^{A_2}}$ is independent from the propagation attribute passed from A_1 to G-Profile. It is a boolean predicate of the form $(\kappa)_2^{t_1^{A_2}}$ = *time past between subsequent modifications* $\leq$ *2ms*.

This way, we can define a boolean function that we will call *mapping activation function f* acting on $(\kappa)_1^{t_h^{A_j}}, (\kappa)_2^{t_h^{A_j}}, \dots, (\kappa)_{n_{t_h}}^{t_h^{A_j}}$, formally

$$f\left((\mathrm{K})^{t_h^{A_j}}\right) = f\left((\kappa)_1^{t_h^{A_j}}, (\kappa)_2^{t_h^{A_j}}, \dots, (\kappa)_{n_{t_h}}^{t_h^{A_j}}\right) \rightarrow \{0,1\}$$

enabling the propagation of a change on $s_g^{A_i}$ to $t_h^{A_j}$ using $m_k^{A_i,A_j}$ if

$$f\left((\kappa)_1^{t_h^{A_j}}, (\kappa)_2^{t_h^{A_j}}, \dots, (\kappa)_{n_{t_h}}^{t_h^{A_j}}\right) = 1$$

Procedure

- A modification occurs on $s_g^{A_i} \in S_l^{A_i}$;
- 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;
- G-Profile verifies the mapping function established on $s_g^{A_i}$ and finds the corresponding $t_h^{A_j}$;
- G-Profile asks the application A_i for complementary data if the target object $t_h^{A_j}$ is involved in a matching function needing additional data;
- 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 source application A_i;
- 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 $t_h^{A_j}$.

3.3.4 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 $\boldsymbol{M}^{A_i,A_j}$. 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 $m^{A_j,A_{j'}}$, $j \neq j'$, 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 $A_j, A_{j'}, A_{j''}, \dots, A_{j'^{\cdots\prime}}$. 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? (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).

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 $f\left((\kappa)_1^{t_h^{A_j}}, (\kappa)_2^{t_h^{A_j}}, \dots, (\kappa)_{n_{t_h}}^{t_h^{A_j}}\right) = 0$. This way, it is impossible to have two modifications on the same target object in 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 $s_{g''}^{A_i}$, propagating to $t_h^{A_j}$) occurred at time t? This might happen due 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.

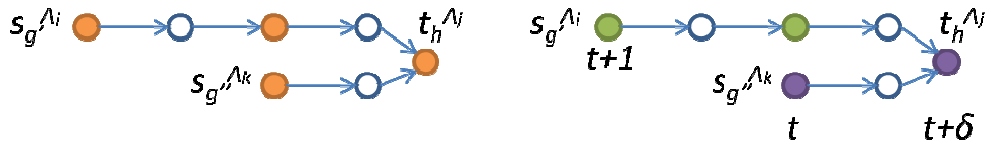

Figure 2. Data propagation drawbacks connected to parallelism.

3.4 A Concrete Example

In this Section we briefly illustrate an example clarifying the use of G-Profile in a multi-application environment. Let us consider the following four applications: A_1 = Facebook, A_2 = Windows Live, A_3 = Amazon and A_4 = eBay. Each application detains specific attributes (we have extracted for each application the most significant ones).

We developed a test environment to illustrate how applications can establish partnerships between them. We assume that, for each application, some possible mapping functions associated to data are already registered in G-Profile. For each couple of applications (A_i, A_j),

$i \neq j$, their administrators gain access to A_i and A_j in order to concretely relate a source set in A_i to a target object in A_j via a specific chosen mapping function.

Figure 3 illustrates the example of the choice of the mapping function $directCopy$ (between a list of predefined mapping functions) relating the attributes $birthday \in A_1$ (Facebook) and $birth\_date \in A_2$ (Windows Live). It is possible, depending on the involved attributes, to choose other kinds of mapping functions, acting differently depending on the structural and semantical properties of the attributes themselves (e.g., the mapping function $append$ between the attributes $employer$ and $company$).

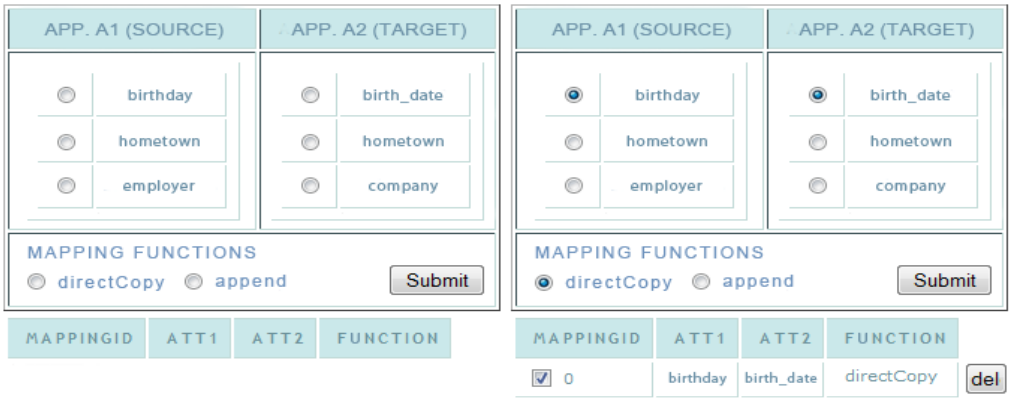

Figure 3. Choice of the $directCopy$ mapping function between A_1 and A_2.

Figure 4 better explains the possibility to choose different mapping functions between different attributes. It shows the result of the application of our technique in the definition of mappings between the four applications in the current example. As we can see, applications in the graph are connected via mapping functions. Each mapping function is denoted by the name of the target application detaining it, followed by the name of the target attribute involved in the mapping and by the name of the mapping function acting on the attribute. E.g., $AMAZON: first\ name: concat$ and $AMAZON: last\ name: concat$ for the mapping functions connecting the attribute $AMAZON: full\_name$ with the attributes $FACEBOOK: first\_name$ and $FACEBOOK: last\_name$.

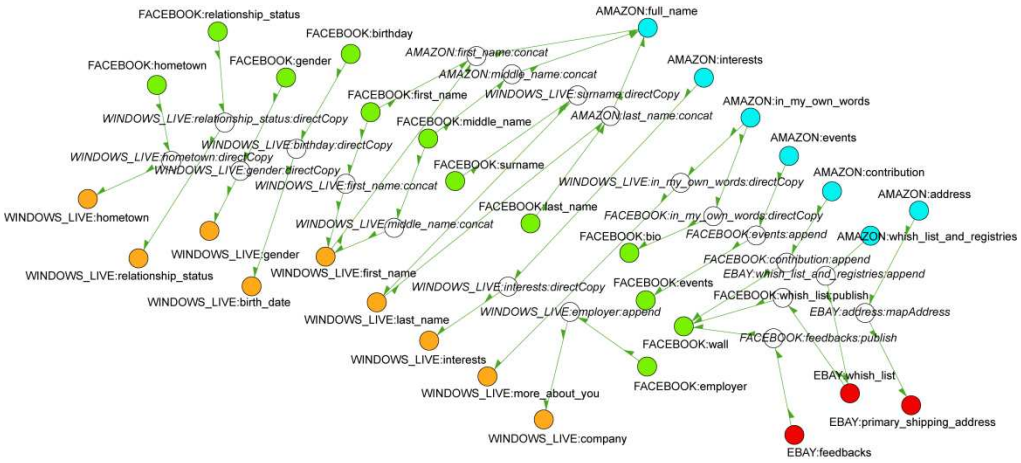

Figure 4. The graph representation of mappings between applications.

Once obtained the mapping functions graph, it is possible to propagate modifications on user data among applications. Due to the importance of this aspect in our technique, more detailed examples will be given in the next section, also providing the evaluation of our work.

4. EVALUATION

In this section we describe the prototype we have implemented and the obtained results. We evaluate the advantages of using G-Profile in user data propagation in terms of *number of incoherencies* generated by user data changes in a multi-application environment. Each modification of an attribute produces, over the time, possible incoherencies in user data belonging to other applications referring to the modified user data. As we have explained at the beginning of the paper, each user manages different user profiles in different applications and she usually manually modifies the same data in different applications. Before the user has updated all her data, they remain inconsistent for a certain period of time. We demonstrate that, given the same period of time, the use of G-Profile leads to a smaller number of incoherencies in the environment. A modification generated on a user data on an application is automatically propagated to the "connected" user data managed by other applications.

4.1 Mapping Graph Generation

For the mapping graph generation phase, our prototype is based on the definition of *applications*, *attributes* and *mappings* between them. Our API has been developed in Java and the generation of these objects has been done automatically. We leave open the possibility to act on $applicationCount$: the *number of applications* in the system and, *for each application*:

- $minAttributeCount$: the *minimum number of attributes*;
- $maxAttributeCount$: the *maximum number of attributes*;
- $minTargetAttributeCount$: the *minimum number of target attributes*;
- $maxTargetAttributeCount$: the *maximum number of target attributes*;

- $minAppToMapWithCount$: the *minimum number of applications to map with*;
- $maxAppToMapWithCount$: the *maximum number of applications to map with*.

This way, each application manages a different number of attributes. Furthermore, each application establishes mappings with a different number of applications and a different number of attributes is mapped depending on the specific application.
For the sake of simplicity, we consider for the moment only 1:1 mappings. A mapping is established between two attributes belonging to two different applications. We suppose that mappings are possible only between semantically related data (e.g., a "postal code" in an application can be mapped to the "zip code" in another one).

We represent our attributes as vertexes of a graph, as illustrated in Section 3.3.2. We indicate each vertex with the letter V, followed by the attribute progressive identification number and the ID of the application detaining it (e.g., the second attribute of the application 1, will be denoted as $V2(1)$).

A mapping between two attributes is established via a mapping function, represented as a function node in the graph. Each function node is denoted by the letters FN followed by a progressive ID assigned by the system.

Figure 5 illustrates the graph obtained with the following parameters: $application\text{-}Count: 10$, $minAttributeCount: 5$, $maxAttributeCount: 10$, $minTargetAttributeCount: 3$, $maxTargetAttributeCount: 6$, $minAppToMapWithCount: 3$, $maxAppToMapWith\text{-}Count: 6$.

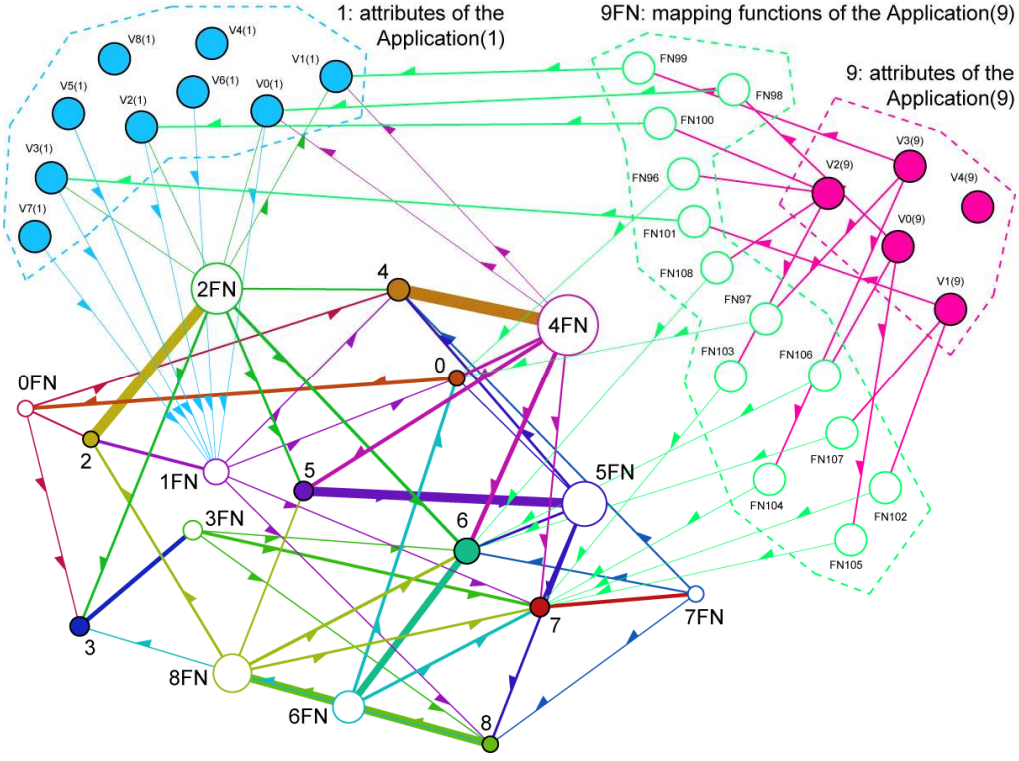

Figure 5. Mapping graph. The diameters of the grouped nodes represents the cardinality of the groups.

For the sake of legibility, vertexes and function nodes have been regrouped based on applications. Only the attributes belonging to applications 1 and 9, and the mapping functions belonging to the application 9 have been expanded.

4.2 Data Propagation Evaluation

Once the mapping graph has been generated, we effectively evaluate the impact of data propagation in a multi-application system. Our evaluation is based on the concept of *modifications* occurring each *period of time* on certain *attributes* in the system in a given discrete *time*. For the sake of simplicity, *user data anonymization* has been e effectuated. Not considering the presence of users does not affect, for the specific implementation, the results of the evaluation.

G-Profile and non-G-Profile Simulated Behavior More specifically, we simulate G-Profile and a non-G-Profile behaviors over a set of applications. In both cases random attribute node modifications occur over a given time. We suppose that the attributes that are directly or indirectly mapped, according to G-Profile, with modified ones become incoherent. In the case of the G-Profile-aware simulation, at each cycle we propagate the modifications to the directly

mapped attributes making them coherent. If no mapping among attributes is provided (i.e., the non-G-Profile-aware simulation), we set random updates of the incoherent nodes, representing the manual change of these attributes over the time. The parameters we can act on are:

- $modificationCount$: the *number of modifications* taking place each *modification time period*;
- $interModCycleCount$: the *duration of the modification time period* in terms of cycles (each modification time period can last one or more cycles);
- $timeStampCount$: the *duration of the simulation*.

4.2.1 Tuning Parameters

Poisson Distribution In order to be more realistic, the number of modifications can be updated at each modification time period following a *Poisson distribution*: $f(k,\lambda) = \frac{\lambda^k e^{-k}}{k!}$. We can act on the $poissonMean$ parameter representing the λ property of the distribution. This way, we represents the fact that the *number* of modifications in a system varies in intensity during the time.

Even the duration of the modification time period can be updated after each modification time period following a Poisson distribution. The involved parameter is called $cyclePoissonMean$. In this case, we represent the fact that the *frequency* of modifications varies in intensity during the time.

Zipf's Law Within the chosen number of modifications in a given modification time period, it is possible to choose the attributes to modify in a non-random way, accordingly to the *Zipf's law*. In probability theory and statistics, it refers to a class of *discrete probability distributions*. The Zipf's law assigns, over of a population of n elements, a probability proportional to $1/r^s$ (s being the exponent characterizing the distribution) to elements of rank $r \leq n$ and zero otherwise, with *normalization factor* $H_n = \sum_{r=1}^{n} 1/r^s$, the n-th *harmonic number*. Formally, $f(r;s,n) = \frac{1/r^s}{\sum_{r=1}^{n} 1/r^s}$. This law is used to describe phenomena where large events are rare, but small ones quite common. In our evaluation, we use Zipf's law to describe the fact that a *large*

number of attributes nodes is modified only *occasionally* whilst *few* attribute nodes are modified *frequently* [1]. We can act on the $zipfSkew$ parameter, corresponding to the s value.

4.2.2 Setting Parameters

Concerning the generation of the mapping graph, our evaluation in terms of incoherencies is based on the following parameters: $applicationCount: 500$, $minAttributeCount: 10$, $maxAttributeCount: 20$, $minTargetAttributeCount: 8$, $maxTargetAttributeCount: 12$, $minAppToMapWithCount: 10$, $maxAppToMapWithCount: 20$.

Concerning the propagation phase, parameters are set as follows: $modification\text{-}Count: 10$, $interModCycleCount: 5$, $timeStampCount: 100$.

Figure 6 shows the positive effects of the use of G-Profile in the lowering of the number of potential incoherencies in a multi-application environment due to modifications in user profiles over a period of time. In particular, in (a) we show the effects of G-Profile with $poissonMean = 10.0$ and $zipfSkew = 0.5$. In (b) we set a higher duration for the modification time period: $interModCycleCount = 10$. In (c) we vary the duration of each modification time period setting $cyclePoissonMean = 10$. Figure 6 (d) shows the superposition of the previous ones.

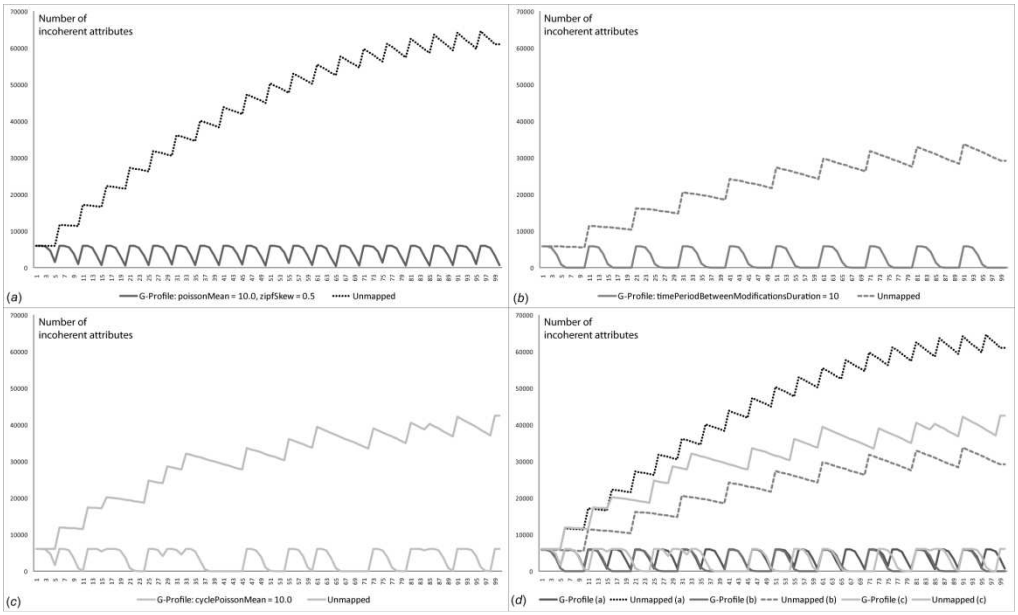

Figure 6. Data propagation effects with and without G-Profile.

In all the configurations, emerges that using G-Profile lead to a quasi-complete incoherence zeroing depending on the $interModCycleCount$ value chosen. A bigger value allows to have a lower value of incoherencies and vice versa.

In particular, independently from the chosen duration of the modification time period, G-Profile shows, applied to the specific simulation, a number i of incoherencies always 0<i<700, while in non-G-Profile environment, the number of incoherencies is continuously growing.

5. CONCLUSIONS

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.

G-Profile aims to be a flexible multi-application user modeling system, able to address typical problems of collaborative distributed environments and in particular to guarantee evolutivity and security and privacy issues in multi-application personalization, aspects that have not been sufficiently considered up to now.

In this paper we have illustrated our technique, with particular reference to the process of user profile data propagation. We have shown that our technique leads to improve the efficiency in the management of user profiles, by lowering the number of incoherencies of user data in multi-application environments.

Our aim for the future is to complete our work by improving our prototype application in order to provide a complete evaluation or our work (taking into account cost models, based on different kinds of data incoherencies); taking into account the user in the different phases of mapping establishment, data propagation, evaluation; introducing the formalization and the methodology for taking into account security and privacy issues and users' expectations.

REFERENCES

[1] L. A. Adamic and B. A. Huberman. Zipf's law and the Internet. *Glottometrics*, 3, 2002.

[2] G. Amato and U. Straccia. User profile modeling and applications to digital libraries. In ECDL '99: Proc. of the 3rd European Conference on Research and Advanced Technology for Digital Libraries, pages 184 – 197, London, UK, 1999. Springer.

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

[4] S. Berkovsky, D. Heckmann, and T. Kuflik. Addressing challenges of ubiquitous user modeling: Between mediation and semantic integration. In *Advances in Ubiquitous User Modelling*, volume 5830 of *LNCS*, pages 1 – 19. Springer, 2009.

[5] S. Berkovsky, T. Kuflik, and F. Ricci. Mediation of user models for enhanced personalization in recommender systems. *User Modeling and User-Adapted Interaction*, 18(3):245 – 286, 2008.

[6] S. Berkovsky, T. Kuflik, and F. Ricci. Cross-representation mediation of user models. *User Modeling and User-Adapted Interaction*, 19(1-2):35 – 63, 2009.

[7] D. Billsus and M. J. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2-3):147 – 180, 2000.

[8] 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*, volume 104 of *CEUR Workshop Proceedings*, 2004.

[9] F. Carmagnola and F. Cena. User identification for cross-system personalisation. *Journal of Information Science*, 179(1-2):16 – 32, 2009.

[10] 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.

[11] 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*, pages 95 – 102, 1999.

[12] P. Dolog and M. Schäfer. A Framework for Browsing, Manipulating and Maintaining Interoperable Learner Profiles. In *User Modeling*, volume 3538 of *LNCS*, pages 397 – 401. Springer, 2005.

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

[14] J. Fink and A. Kobsa. A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web. *User Modeling and User-Adapted Interaction*, 10(2-3):209 – 249, 2000.

[15] G. H. L. Fletcher. The Data Mapping Problem: Algorithmic and Logical Characterizations. In *ICDEW '05: Proc. of the 21st International Conference on Data Engineering Workshops*, page 1263, Washington, DC, USA, 2005. IEEE Computer Society.

[16] 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

[17] 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.

[18] 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 *ITS '98: Proc. of the 4th Int. Conference on Intelligent Tutoring Systems*, pages 494 – 503, London, UK, 1998. Springer.

[19] U. Hansmann, L. Merk, M. S. Nicklous, and T. Stober. *Pervasive Computing: The Mobile World*. Springer Professional Computing. Springer, August 2003.

[20] D. Heckmann and A. Krueger. A user modeling markup language (UserML) for ubiquitous computing. In *User Modeling*, volume 2702 of *LNCS*, pages 393 – 397. Springer, 2003.

[21] 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.

[22] D. Heckmann, T. Schwartz, B. Brandherm, M. Schmitz, and M. von Wilamowitz-Moellendorff. GUMO: The General User Model Ontology. In *User Modeling*, volume 3538 of *LNCS*, pages 428 – 432. Springer, 2005.

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

[24] A. Kobsa. Generic user modeling systems. *User Modeling and User-Adapted Interaction*, 11(1-2):49 – 63, 2001.

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

[26] A. Lorenz. Agent-Based Ubiquitous User Modeling. In *User Modeling*, volume 3538 of *LNCS*, pages 512 – 514. Springer, 2005.

[27] G. I. McCalla, J. Vassileva, J. E. Greer, and S. Bull. Active learner modelling. In *ITS '00: Proc. of the 5th International Conference on Intelligent Tutoring Systems*, pages 53 – 62, London, UK, 2000. Springer.

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

[29] B. Mehta, C. Niederée, A. Stewart, M. Degemmis, P. Lops, and G. Semeraro. Ontologically-Enriched Unified User Modeling for Cross-System Personalization. In *User Modeling*, volume 3538 of *LNCS*, pages 119 – 123. Springer, 2005.

[30] 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.

[31] J. Park and S. Ram. Information systems interoperability: What lies beneath? *ACM Transactions on Information Systems*, 22(4):595 – 632, 2004.

[32] 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*, pages 1 – 7, New York, NY, USA, 2009. ACM.

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

[34] L. Razmerita, A. A. Angehrn, and A. Maedche. Ontology-based user modeling for knowledge management systems. In *User Modeling*, volume 2702 of *LNCS*, pages 213 – 217. Springer, 2003.

[35] 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

[36] J. Vassileva. Distributed user modelling for universal information access. In *Universal Access In HCI: Towards an Information Society for All, Proc. of HCI International '01*, volume 3, pages 122 – 126, New Orleans, USA, 2001. Lawrence Erlbaum.

[37] J. Vassileva, G. Mccalla, and J. Greer. Multi-agent multi-user modeling in i-help. *User Modeling and User-Adapted Interaction*, 13(1-2):179 – 210, 2003.

[38] M. Viviani, N. Bennani, and E. Egyed-Zsigmond. G-Profile: a Multi-application Personalization System. In *Proc. of ERATSI Workshop*, in conjunction with INFORSID, May 2010.