Multimedia Systems (2009) 15:187–199
DOI 10.1007/s00530-009-0162-4

REGULAR PAPER

# Distributing the personal digital environment throughout your entertainment environment: handling personal metadata across domains

**Artur Lugmayr · Simon Reymann · Volker Bruns · Jakub Rachwalski · Stefan Kemper**

Published online: 19 June 2009


**Abstract** It is a fact, that we are surrounded by more and more ubiquitous services in entertainment computation. All sorts of devices are more and more connected, and personalization becomes more and more a major issue. We are living in a world with two layers: the real physical layer being our real-world, and its synthetic overlay consisting of location-based services, chatting applications, or Web 2.0 offers. In the real-world, communication between humans is a matter of personality of different persons: exchanging information about each other, finding common interests, or finding common conversation themes. In the synthetic overlay, communication becomes a matter of distribution of personal profiles or automating profile matching. The open-source platform Portable Personality (P2) (www.portable-personality.org) faces this challenge, and provides a platform for cross-service interchange of personal context information based on any generic metadata type. P2's software architecture is designed for mining, enriching, and exchanging personal profiles between arbitrary multimedia services. The long-term vision of P2 is to provide a personality profile rather than a personal context information profile to enable communication between human and device as a matter of personalities, rather than automated matching of profiles. Two different scenarios, the Smart Social Network (SSN), and the personalization of audio-visual data are presented as practical use-case for P2.

**Keywords** Ambient media · Pervasive computation · Personalization · Personal profile · Context awareness · Metadata · Context profile · Ubiquitous computation · Portable personality · P2 · Personality profile

A. Lugmayr (✉) · S. Reymann · V. Bruns · J. Rachwalski · S. Kemper
Tampere University of Technology, Korkeakoulunkatu 1,
P.O. Box 553, 33101 Tampere, Finland
e-mail: Artur.Lugmayr@tut.fi; lartur@acm.org
URL: http://namu.cs.tut.fi

S. Reymann
e-mail: Simon.Reymann@tut.fi

V. Bruns
e-mail: Volker.Bruns@tut.fi

J. Rachwalski
e-mail: Jakub.Rachwalski@tut.fi

S. Kemper
e-mail: Stefan.Kemper@tut.fi

## 1 Introduction

The provision of a general valid and generic personality profile is gaining of more and more importance in entertainment services. The personality profile shall include information about the usage-context of media services and contain information about the personality of the consumer. These media services can be arbitrary and range from simple playlist compilations of audio-visual content to complex feedback profiles in interactive television or other profiles coming from prominent web-services such as YouTube, IMDb, and lastFM.

This research paper provides an overview of a metadata-based solution for the management of personalized content. Multiple applications, such as audio/video players contribute with user-context information enabling future personalization scenarios. A metadata specification in form of XML is presented providing a generic container format for representing a personality user profile. To enable generality, the personality profile simply extends existing XML standards by adding attributes and thus keeps the existing structure intact. This enables the creation of slim and resource-efficient personality schemas. These are well suited for a deployment on resource-sensitive client devices such as mobile phones or embedded entertainment devices such as STBs.

The consumer shall have access to an ubiquitous personalized environment anytime, anywhere, and anyhow.

Let us consider P2 in the context of existing web-services, such as YouTube, IMDb, or lastFM. Each single service collects consumer data, and a remarkable amount of personal information from each consumer. P2 acts as metadata harvester, and obtains any type of personal profile from these services. These data are compiled to a personality profile and provided to any other application in need for personalization information. P2 collects, e.g., music preferences from a single person from lastFM, is fed by YouTube's video preference parameters, or simply obtains the contacts on one's Skype contact list. Thus, if the consumer enjoys rock music on lastFM, it is very likely that he buys a rock music CD in a music shop. Thus, the data that P2 provides, e.g., to the shop advertisement displays, contain the favorite music genre. As P2 handles any type of metadata, the application areas and integration of other web-services provides many possibilities. A basic overview of P2 is depicted in Fig. 1.

Portable Personality (P2) tackles the following challenges:

- integration of manifold service specific metadata formats for the storage of personal profiles to one P2 profile;
- exchange of metadata across ambient networks, devices, and services to achieve a seamless service space rather than single services;
- provision of an advanced mining and personalization algorithm for smartly mining personal profiles;

The article concludes with the presentation of two scenarios, in the social networking domain, and in the audio-visual domain.

### 1.1 Application context of P2

P2 is a novel architecture for personality profile handling in a distributed network. Existing solutions rely on a centralized server for handling identity and personal profiles. The reason for this is simple: rather than providing consumer-data-hungry service provider with personal data, consumer data shall be in the hand of the consumer for his own benefit. Where OpenID or the Liberty Alliance [15,19] provides server-side digital identity management for user authentication in a seamless and trusted way, P2 relies on the handling of *personality profiles* rather than identity. Each user identifies himself to the application, rather than relying on third party authentication technology and has full control: as to which data are private and which public. To integrate services into an ambient space of loosely connected devices, P2 relies on XML based metadata. Thus, rather than defining a single, rigid metadata schema, P2 offers a pure personality-based metadata management architecture. Existing proprietary applications, such as Skype, YouTube, Flick, or TV-Anytime provide their data to P2. The strength of the application, therefore, is how to manage and distribute personality profiles based on generic 'input' metadata. This metadata is forwarded to client applications requiring specific types or similar types of data. P2 makes use of the efforts as undertaken by the W3C [32] by providing annotations, and tagged information in a consumer-personalized way.

### 1.2 P2 as open source framework

The current implementation of P2 consists of two software packages or layers (see Fig. 2): first, the P2 Core layer, which includes the functionality for metadata handling of P2 profiles (see e.g. [4]); and second the AmbiNET layer, implementing a heterogenous solution for transport of personality and context profiles across networks and devices (see e.g. [24]).

Figure 3 presents an example scenario of the Portable Personality (P2) environment. The stationary domain represents the home television environment (e.g. set-top-box). The P2 Provider collects the consumer data (e.g. watched TV program) and annotates it with P2 data to provide it to the P2 Daemons in a computer-readable format. The P2 Daemons are also responsible to distribute the data to other domains, such as onto the mobile phone. If the consumer enters a new stationary domain, the P2 profile can be synchronized with the equipment of the new stationary domain. However, the P2 daemons also can enrich the profile to generate a higher level personality profile, as well as they are responsible for data archiving. A more detailed practical example will be presented at the end of the article.


**Fig. 1** *Portable Personality (P2)* software framework with one unified personality profile for the exchange between domains

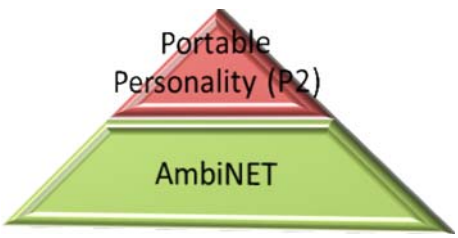


**Fig. 2** The two layers of the *Portable Personality (P2)* software framework responsible for performing processes on personal profiles including the underlying ambient network for personal profile distribution

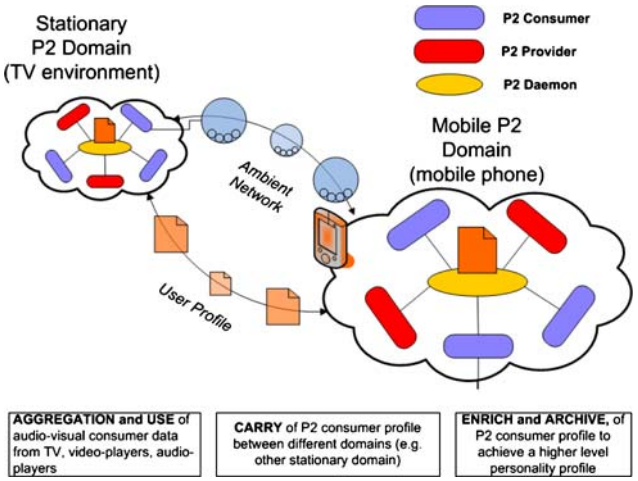


**Fig. 3** Application of the Portable Personality (P2) environment in the case of television—the stationary P2 domain is represented by the TV environment (e.g. set-top-boxes, PC video player). The mobile P2 domain is represented via a mobile phone or PDA transferring the P2 profile between different domains (e.g. between living room and bedroom or the friends' home)

### 1.3 Related work

Metadata on general level is based on XML as description language, developed by the W3C [31,33–35]. However, the more interesting standards are domain-oriented. There exist many specific metadata standards for specific domains, which are also described within the *Dublin Core Metadata Initiative (DCMI)* [5] specifications or the *Resource Description Framework (RDF)* [31]. One example is the *Friend of a Friend (FOAF)* project, specifying links between people including their descriptions and social networking information [28]. *Description of a Career (DOAC)* [6] metadata provides an RDF metadata vocabulary for describe CV-like data for professional capabilities of an employee [6]. Embedding of various types of file metadata into files of some common formats is enabled with the *Adobe Extensible Metadata Platform (XMP)* [1]. However, not keeping out of view are metadata formats not based on XML, such as IPTC [10] headers to annotate digital photographs with EXIF data. ID3 tags are an example for annotating audio files with associated metadata [9]. From the mathematical and algorithmic side, personalization is discussed from the MPEG-7/MPEG-21 perspective in [29]. A distributed platform for personalization is presented in [8] and a TV-oriented framework in [3]. An excellent work for personalized content on mobile phones is [13], discussing several aspects of mobile content personalization and its metadata handling. A few other works worth reading, which can give insights into human behavior and personalization are [2,3,7,14,16,17,20,21].

Several mentioned metadata standards lack on a cross-domain interoperable software framework, which allows annotations similar to folksonomies on the web, where each user can use any type of metadata to tag content. P2 enables handling of fuzzy metadata; thus any kind of input metadata and can be considered as general management platform of personal profiles. Publications introducing and conceptualizing P2 can be found in [4,22–24]. The open-source forum can be found on www.portable-personality.org. The strength of P2 lies especially in the capability to be able to work with any type of XML-based metadata and flexibility. Despite other synchronization languages such as, e.g., SyncML [27], P2 provides a thin middleware architecture with all its advantages. It also differs from simple personalization standards as TV-Anytime, as P2 is cross-domain applicable. Other works related to ambient intelligence on general level can be found in [11,12,25].

## 2 System architecture

The principle of P2 is based on the abstract architecture presented in Fig. 4. Personal context profiles are managed by daemons. Daemons are services running in the background and perform tasks related to metadata management (e.g. storage, synchronization, mining, profile distribution). In the context of P2 they are called *P2 Services* and are responsible to manage context metadata coming from providers.

*P2 Provider* are software programs responsible to harvest metadata from arbitrary sources. These can be implemented as plug-ins for music players, instant messaging software, or video players. P2 Providers feed the P2 Services with annotated context profiles for further processing. Context profiles are requested by *P2 Consumers*. They are responsible to request the data and trigger actions based on the data provided by daemons. Daemons pre-prepare the related metadata in such a form, that the consumer software is capable of interpreting this data. One example scenario is to provide a video-player (consumer) with the metadata related to music taste. This scenario is described in further detail in later sections of this paper. Different from other stationary systems, such as YouTube or Google, personal data are distributed


**Fig. 4** Overview of the P2 system architecture with multiple provider and consumer nodes interconnected through the distributive P2 daemon network

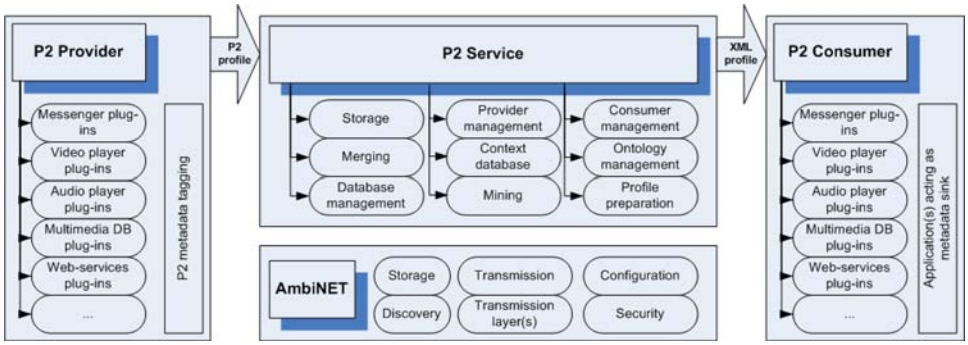


**Fig. 5** Overview of the different components of the P2 personal context profile management software

among the consumers' own devices, rather than stored on a central server. The consumer can carry his personal profile to different locations, where it is exchanged according different levels of data privacy. Within the scope of this section, the different components of the system architecture are described in further depth.

### 2.1 Components of the P2 framework

A more detailed description of the components of the P2 framework is given in Fig. 5. The main components of the framework are

- *P2 Provider:* provision of metadata tagged with P2 conform attributes;
- *P2 Service:* daemon for the management of metadata;
- *AmbiNET:* network component for distributed profile transport;
- *P2 Consumer:* data sink and application(s) for triggering actions;

The overall functionality of each of the components of the framework is explained in further depths within the scope of this section. The main component of the framework is the personal profile as such, which is as well explained in further depth in later sections.

### 2.2 P2 provider

P2 offers an API for the development of providers. This API is independent from provider type, and responsible for providing a data interchange interface for metadata handling with P2 services. As provider is implemented within the scope of certain applications, they are aware about the type of data (e.g. sensor metadata, person metadata, playlists). Typical for metadata is its ambiguous notation, and many standards exist expressing similar or the same data. To face this challenge, two mechanisms have been implemented: first, annotation of metadata with ontology vocabulary by providers as, e.g., based on the available schemas on [26]; and second, annotation of metadata with rules for merging metadata by P2 services. This makes metadata machine processable. One other aim is to keep the original metadata intact, that it can be used for the same type of application and to reduce metadata transformations. Therefore, we selected these two mechanisms.

As illustrative example, let us consider *Friend of a Friend (FOAF)* [28] metadata. Buddy lists from instant messenger can be expressed as FOAF metadata, as we know that buddies are friends of the person using the client. Thus, buddies can be tagged as friends. The main aim is to keep original intact, and it can be fed to another messenger client in its original form. The practical use is illustrated in Table 2, and 3. Table 1 shows the original metadata coming from a favorite list on a music player. Table 2 shows the tagged metadata with merge keys. Table 3 shows metadata tagged with merging rules. Each required management function has its own P2 attribute defined. The attributes' values consist of coma separated fields (CSF) defining options for a single personal content item of the original XML content. If the attribute refers to a couple of elements each field is separated by a semicolon. The following attributes types are valid:

- Single (S SYNTAX), e.g., p2:item- with 1 VALUE FIELD or V FIELD;
- Comma separated pairs (CSP SYNTAX), e.g., p2:merge- with two fields in CSF;

**Table 1** Original metadata coming from a video application

```
<favourites>
  <show>
    <title>Found</title>
    <director>Spielberg</director>
    <value>2.5</value>
  </show>
  <show>
    <title>Spiderwoman</title>
    <director>Spielberg</director>
    <value>3.3</value>
  </show>
</favourites>
```

**Table 2** P2 profile annotated with specific P2 attribute extensions

```
<favourites p2:item="1">
  <show p2:item="1" p2:limit="title,1,9">
    <title p2:merge="title,1">Found</title>
    <director
p2:merge="director,1">Spielberg</director>
    <value p2:merge="value,3">2.5</value>
  </show>
  <show p2:item="1" p2:limit="title,1,9">
    <title
p2:merge="title,1">Spiderwoman</title>
    <director
p2:merge="director,1">Spielberg</director>
    <value p2:merge="value,3">3.3</value>
  </show>
</favourites>
```

**Table 3** P2 personal profile metadata tagged with merging rules

```
<foaf:nick       p2:mergemodes="*foaf:nick,
IFDIFFERENT_INSERTAFTER_TAKEOLDER"
p2:sourcekeys="*foaf:nick,1">
Lartur
</foaf:nick>
<favourites p2:item="1">
```

– Comma separated thirds (CST SYNTAX), e.g., p2:limit- with three fields in CSF;
– Comma separated quads (CSQ SYNTAX), e.g., p2:delete- with four fields in CSF.

The first item in a CSF is called REFERENCE (R) FIELD, the second MODE (M) FIELD, the third ADDITIONAL (A) FIELD, and the fourth EXTRA (E) FIELD. A P2 attribute placed in an element with an R FIELD may refer to the element, its value, or a child element. However, if it is an attribute it contains the name of the attribute proceeded by $, e.g., $<attribute_name>. What is more, the default value of P2 attributes may be changed for the element and all its values, attributes and the child nodes. In this case the element is preceded by @, e.g., @<element_name>.

### 2.3 P2 service

The core of the P2 framework is the P2 Service. The service can be described as metadata management platform or metadata engine. Its main responsibility is to manage and merge metadata coming from various providers into a personal context profile. Personal context profiles act as data source which need to be enriched to be useable by a consumer application (e.g. transformation of metadata, mining of metadata). In its current form, metadata information is stored in one XML based profile. An extension towards a tiny metadata repository is currently under development, to enrich the capabilities for data mining, retrieval, and search. The central tasks of the P2 service relate to the management of the life-cycle of personal profiles, which are described in a later section of this article in further depth.

### 2.4 AmbiNET

The AmbiNET component is responsible for the exchange of personal profiles on transport level and enables the communication between several components of the framework. AmbiNET is independent from transmission layer, thus its protocol stack is independent from the underlying physical network. AmbiNET allows the exchange of information of a various set of transmission technology, such as Bluetooth, IP-networks, Internet, infrared, or WiFi. Transmission of profiles is seamless; thus the configuration of the transmission method is according the available network. On the network level, P2 distinguishes between the following domain-specific network types:

– *global domain:* exchange and management of personal profiles across networks via daemons;
– *local domain:* exchange and management of personal profiles on a specific device between applications;
– *cross-domain:* exchange and management of personal profiles between domains;

Each application component feeds its metadata to the P2 service component. Each P2 service component sits on top of an AmbiNET implementation, responsible for distributing the personal context profile to the registered other P2 services. The utilized protocol stack is arbitrary, and various connecting nodes can be managed. An overview is given in Fig. 6.

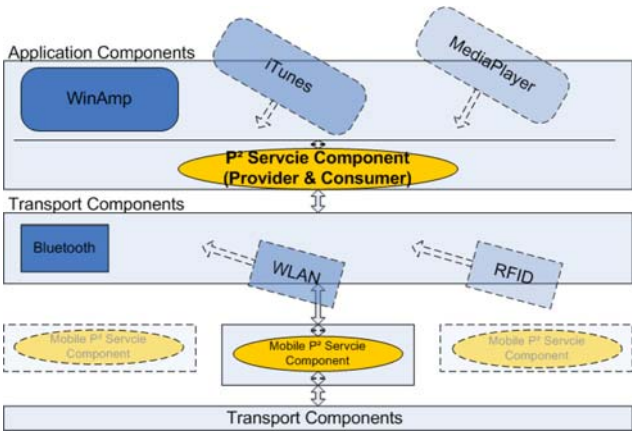


**Fig. 6** Overview of the P2 middleware architecture and its modules

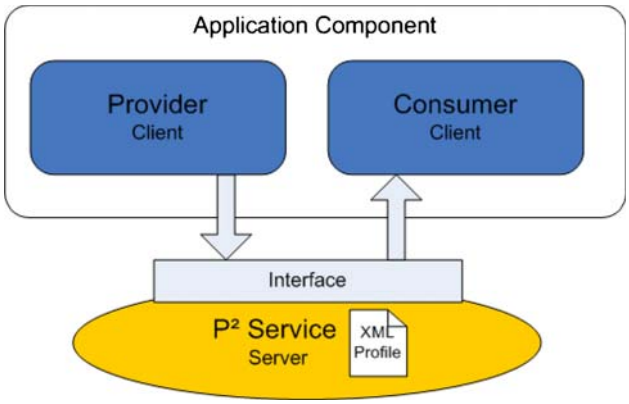


**Fig. 7** Local domain consisting of the triple P2 Consumer(s), P2 Provider(s), and P2 Service on one single device

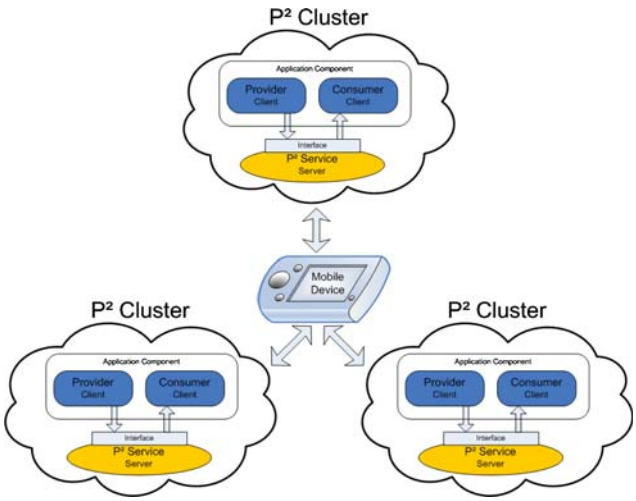


**Fig. 8** AmbiNET as network to interconnect P2 cluster of local domains

A local domain consists normally of the triple P2 Consumer(s), P2 Provider(s), and P2 Service on one local device. The information exchange is cross-application boundaries by providing a seamless metadata information exchange. The general application of P2 in a local domain is illustrated in Fig. 7.

The global domain of AmbiNET consists normally of clusters of P2 components interconnected via P2 services running on single devices. AmbiNET is responsible to interchange the data on transmission layer between the various clusters (see Fig. 8).

### 2.5 P2 consumers

As for P2 Providers, an API for P2 Consumer is available. The API abstracts functionality required for the consumption of personal profiles. Due to the wide variety of available potential consumers, a very high abstraction level is required. One of the core ideas is to provide metadata in its original form to the consumers, thus no metadata transformation shall be required. However, P2 services allow enriching metadata profiles, to provide additional functionality to consumers. Two different scenarios for P2 Consumers are introduced within the scope of this article, and described in further depth in form of two different scenarios.

## 3 Personality profile life-cycle

For handling metadata concerning personality and consumer profiling, it is essential to develop a life-cycle for the handling process to identify the functionalities of different system components on an abstract level. These relations are illustrated in Fig. 9. Within the scope of this section, the different life-cycle phases are illustrated. The section also illustrates the annotation process undertaken by the P2 environment.

### 3.1 Personality profile life-cycle

The detailed life-cycle of a personality profile is illustrated in Table 4 and Fig. 9. After profile creation and initialization of the personality profile with the basic user data, the profile can be exchanged between different entities (see [4] for a more detailed description).

#### *3.1.1 Aggregate*

In the creation phase, providers monitor the consumers' actions and harvest any raw metadata in any format which is relevant for the P2 service. P2 providers are capable of collecting any generic type of metadata as, e.g., FOAF or ID3 metadata and annotate it with the essential P2 metadata extensions. Each provider is sitting on top of any type of application or service such as, e.g., video-player, audio-player, social networking client, and online service. The provider is implemented as stand-alone application or service-specific plug-in. The provider is responsible for collecting, creating,

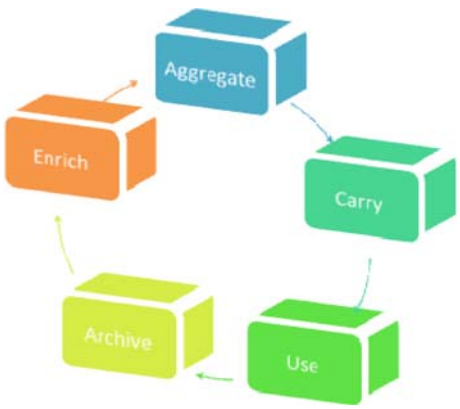


**Fig. 9** Personality metadata profile life-cycle

**Table 4** Description of metadata profile life-cycle phases

| Phase | Functionality | Components | Metadata |
|---|---|---|---|
| *Aggregate* | Collection, creation, and transformation of metadata to make it P2 readable | P2 Providers | Input generic metadata structures (either in XML or non-XML) |
| | | | output annotated native metadata in P2 conform metadata format (in XML) |
| *Carry* | Distribution, carrying, synchronizing, and exchange of the P2 metadata profile | P2 Daemons | P2 conform metadata |
| *Use* | Conversion, utilization, and processing of P2 metadata | P2Clients | input P2 conform metadata |
| | | | output application dependent metadata format |
| *Archive* | Management and storage of the P2 metadata profile | P2 Daemons | P2 conform metadata including profile management metadata |
| *Enrich* | Mining and enriching existing profiles across services | P2 Daemons | P2 conform metadata |

and pre-processing the P2 relevant metadata. The provider annotates the metadata with P2 rules and converts it to a P2 readable format. The strength of the P2 provider is that it is capable of processing any type of metadata and converting it to a P2 format. One example for the aggregation phase is the ID3 tags of audio files. Songs are annotated with information such as title, artist, genre, play counts, or user rating. The latter can be used to calculate a more sophisticating user rating in combination with the file age. P2 providers are capable of converting ID3 tags to XML and annotate it at the same time with P2-relevant annotations.

#### *3.1.2 Carry*

In the carry phase, daemons are responsible to distribute personal profiles across networks and devices. It could be also described as exchange phase of P2's personality profiles but as mobile devices are the predominant device acting as profile carrier, 'carry' seems to be a more adequate description. The P2 profile is distributed between P2 provider and P2 daemons independent of the implemented communication technologies. Configurations ranging from Bluetooth, InfraRed (IrDa), 802.11 Wireless LAN up to wired LAN are possible. P2 realizes an ambient network [36], which is a heterogeneous solution for ambient ad-hoc networks over any type of network channel.

#### *3.1.3 Use*

In the use life-cycle phase, the P2 profile of the consumer is utilized by the P2 consumer. This can happen arbitrarily and is dependent on the functionality of the application. As soon as a client detects a daemon application, the P2 profile can be exchanged. A consumer can either explicitly subscribe to these events, or request a concrete profile. However, the discovery process should happen seamlessly. During this life-cycle phase the consumer becomes aware of the P2 service. In the background different P2 daemons exchange the profiles seamlessly. Stating an example—the consumer arrives at a new domain as, e.g., another TV environment in the sleeping room. The daemons of the application become aware of each other, and start to synchronize and update their profiles to provide a personalized experience.

#### *3.1.4 Enrich*

Currently, the enrich functionality of P2 is still under process and investigation. However, we undertook first case studies and are currently exploring more advanced algorithms for performing this task. The principle idea is the creation of a personality profile, which should represent the personality of the user, rather than simple context usage profiles or media usage lists of the consumer.

### 3.2 Personality profile modeling

Every XML package received by a daemon may be very distant in structure, as it could come from different providers. What is more, each contained personal content item needs to be identified before it is processed by the daemon according to given management rules. In order to process an unknown standard the personal content provider needs to identify a personal item by extending it with specific P2 attributes (from the xmlns:p2="http://www.portable-personality.org/") before sending it to the daemon. If the daemon knows a standard it is able to identify items and extend them automatically by default rules. As the provider or the daemon knows the structure and extensions he can easily create a complaint version of the XML schema accordingly. Furthermore, when sending personal content to a consumer the daemon deletes all P2 attributes generating valid XML content. Known standards, like Friend of a Friend (FOAF)

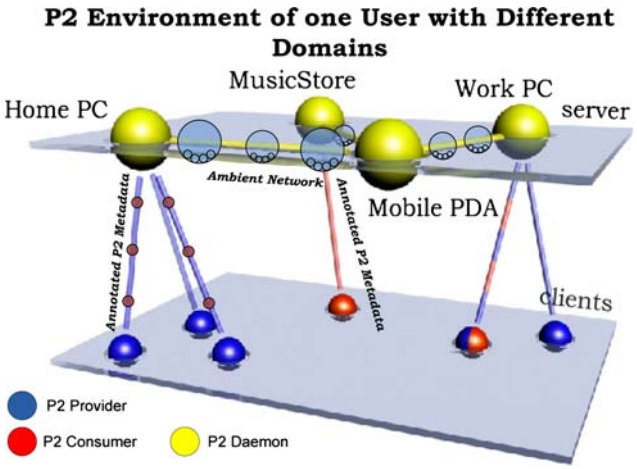


**Fig. 10** Annotated P2 metadata as exchange entity between different domains

[28], Description of a Career [6] or Media RSS could be processed automatically.

P2's context model is based on the ideas of [13], where context is modeled based on the different sources of context information. Context ($C_i$) is modeled as a triple ⟨E, U, D⟩, where E relates to information concerning the environment (e.g. location, time, service), U relates to information about the consumer (e.g. activity, emotions), and D relates to the device context (e.g. sensors, interface). Thus, any information relevant for providing information about the consumer, his context, or his personality will be added to the profile. In extension to [8], P2 allows a more generic context model for the personality profile, where the profile is modeled as tuple of $\langle A_q, S_q \rangle$, where $A_q$ is the set of attributes for a particular media service (e.g. genre), and $S_q$ the score of the consumer of this service. In P2, a particular context $C_i$ is modeled of a set of attributes $\{p_1, p_2 \ldots p_k\}$ across domains, with a relevancy factor of $R_q$ for specific domain contexts. This guarantees a generalization of metadata mining across application domains.

To state an example, let us consider the context 'user is in a music shop'; thus the context can be described as $E_i = \{\text{musicshop}\}$, $U_i$=[19], $D_i = \{\text{musicshopofferings}\}$. On attribute level, the context is modeled as $C_i = \{p_{\text{musicgenre}}, p_{\text{moviegenre}}, p_{\text{mood}}, p_{\text{musicshopofferings}} p_{\text{location}}, p_{\text{friendsmusictaste}}\}$. During the mining operation of the existing data continued in the personal profile, the different context parameters are weighed with $R_q$; thus, e.g., $R_q = \{0.9, 0.4, 0.2, 1, 0.9, 0.2\}$. The example music store is illustrated in Fig. 10.

## 3.3 Personality profile operations

This section of the paper describes the changes of the metadata structure and possible operations on the personality profile, especially merging.

In XML content some of the elements together with their child nodes should be treated as one piece of information (further referred as ITEM). In the exemplary code each "show" element represents an ITEM and is then described by several child nodes like "director" or "title". The element must be marked with the p2:item="1" attribute which is of S YNTAX (using p2:item="0" ignores the element and is considered as default). ITEMs represent an actual personal content item and hold all required rules for handling contained data. In some XML structures, like iTunes XML in which the parent-child relationship is not defined by its structure the usage of p2:key, for parent nodes, and p2:parentkey, for child nodes is invented. With this, parent-child relationships are uniquely identified.

### *3.3.1 Merging*

The profile merging is one of the most important features in the P2 system. In order to merge two ITEMs, the system needs to know what ITEMs are of the same nature, i.e., what ITEMs represent the very same personal content but hold different values. It is important to identify the type of an ITEM before merging it with an existing ITEM. Due to the fact that two ITEMs can be of a different XML structure, an identification and merging attribute is implemented. The identification of an ITEM merge key (identification key) is done by using the p2:merge="⟨REFERENCE⟩,1" attribute which defines ⟨REFERENCE⟩ as an identifier of the ITEM. All values of all defined identifiers and their location within the structure must be equal before declaring two ITEMs to be of the same type. After characterization through the identifiers the ITEM's child elements get merged according to specified merging modes, each of them representing an individual merging strategy. Merging strategies in fact can be of any type, e.g., sum, multiply, take the newer, concatenate etc.

For merging, a daemon must hold information about the provider source and details about its type. The source is provided as a Globally Unique Identifier (GUID) and the type is represented by its namespace. Content of multiple sources (multiple GUIDs) embedded in the same namespace specification is merged into one XML representation. Merging of personal content originating from two different namespaces, instead, is not supported. The final result is a collection of multiple namespaces holding personal content of multiple profiling sources.

### *3.3.2 Sorting, limiting, and deleting*

Due to the fact that the storage capacity of a daemon is finite, e.g., on mobiles, the provider may set rules when to delete an ITEM using additional P2 attributes. The first option could be deleting on limit in which the number of ITEMs contained in the profile could be limited up to a certain value

using the p2:limit attribute (CST SYNTAX). The limitation takes effect on every ITEM which lies at the same location within the XML structure holding the same ITEM element name and having the same p2:limit attribute defined. The M FIELD defines the sorting mode, e.g. ascending or descending. The R FIELD refers to the element on which the sorting is based on, while the A FIELD defines the limitation value, e.g., 100. A second option is deleting on a value by using the p2:delete attribute (CSQ SYNTAX). An ITEM may be deleted when the value of a certain attribute or child element reaches a limit. The M FIELD holds the condition to be met, e.g., =, <, >, <= or >=, the A FIELD states the limit value and the E FIELD provides grouping functionalities meaning that the conditions of all group members need to be met before deleting. With this, logical AND and OR operations are implemented.

## 4 Scenario 1: smart social network

### 4.1 Overview

The idea of the *Smart Social Network (SSN)* scenario is simple and follows the ideas of social network sites. The difference is that the SSN scenario adds locality information to social networking—friends that are friends on social network sites or instant messenger should discover each other in a physical location. The physical location can be arbitrary—university, cafeteria, working place, or in the living room. A SSN server registers consumers coming into the physical location, and matches data with the currently present ones. The SSN scenario can be also described as physical co-located messenger software.

### 4.2 System architecture

An overview is given in Fig. 11. In its current form, SSN information is obtained from instant messenger programs. Instant messenger programs contain a list of friends, which a consumer might like to meet in a specific physical place. Normally instant messengers are installed in a stationary domain, such as PCs. Within the P2 project, plug-ins fro Skype and Microsoft messenger have been developed. These plug-ins deliver the contact data from the messenger software to the P2 service, which is running as daemon on stationary domain devices (e.g. PC). Contact information is encoded via *Friend of a Friend (FOAF)* metadata [28] and annotated by P2 metadata extensions. These extensions define a set of rules, how specific FOAF profiles coming from a data provider can be merged into the global P2 personality profile.

The P2 service is responsible to distribute the global profile between various P2 services running in the network, which belong to this specific consumer. As such, a mobile phone can act as temporary storage for the personality profile in the mobile domain. Therefore, when the consumer is moving with its mobile platform to another service domain, offering a SSN, a service running on the mobile domain platform connects to this domain, and extracts the relevant FOAF information from the global personality profile. The FOAF data containing the social network of the consumer is forwarded to the 'social server', which is responsible to match FOAF data from one specific consumer with the other registered users. If a match is found (e.g. a consumer has another friend on his messenger contact list, who is also physically present), this information is forwarded to the client. Metadata coming from various P2 components is illustrated in Table 5 in Appendix A.

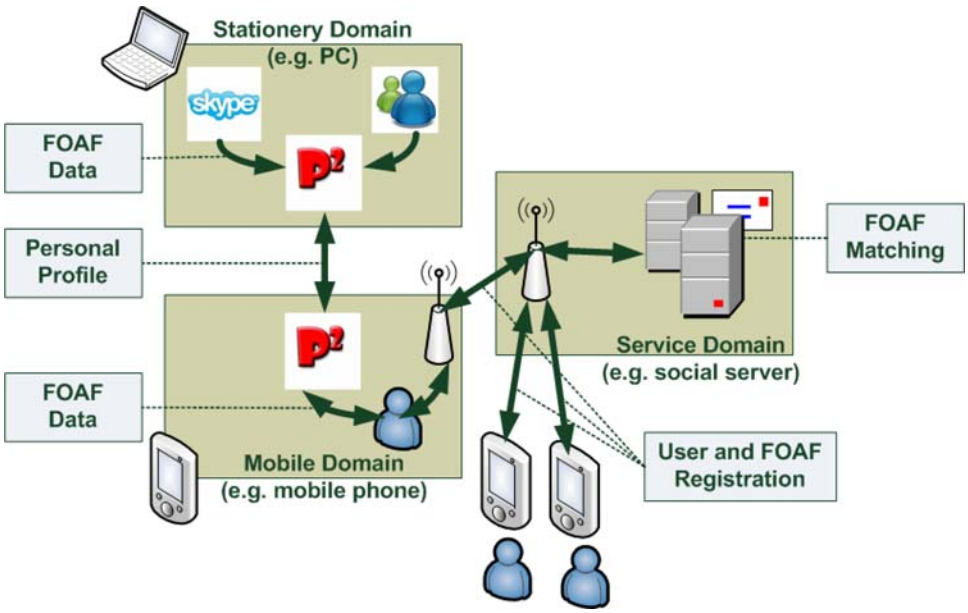


**Fig. 11** Smart Social Network Scenario as location-based messenger software, keeping track of friends on instant messengers and notifying the consumer if one is physically close-by

### 4.3 Implementation specifics: metadata merging of personal profiles

The implementation of the SSN architecture is based on the standard P2 architecture. It consists of several components of the overall P2 framework:

*P2 Provider:* the provider wraps APIs from instant messengers to access FOAF information. This information is annotated by merging rules for merging more profiles coming from different messengers (e.g. extracting of metadata from various clients, annotation via P2 metadata merging rules, and forwarding it to the P2 service);

*P2 Service:* the P2 service is responsible for merging various profiles, and delegating the global personal profile to other P2 services running across various devices (e.g. merging FOAF metadata from skype and Microsoft messenger);

*P2 Consumer:* a P2 consumer is responsible to extract the essential personality metadata from a personal profile and to provide it to a service (e.g. forwarding the FOAF information to the SSN server);

*SSN Server:* utilization of the metadata provided by various users and performing actions based on the input data (e.g. matching FOAF information across users in SSN server proximity and notifying these that friends are close-by);

*AmbiNET:* network platform for the distributed exchange of P2 profiles across any type of networks, as described in previous sections).

## 5 Scenario 2: personalization of audio-visual data

In this section the P2 framework is applied in the context of audio-visual data. Any kind of music player software can act as metadata provider within the P2 framework. MP3 players (in our case Winamp) collects playlists and other kind of consumer data. In this case the implementation is based on a PC environment; thus providers and daemons run on one single PC. After tagging the data coming from a provider with synchronization and type information, the profile is distributed by the daemon to various other daemons. Other daemons can, e.g., be a mobile phone, interconnected with the AmbiNET to the PC daemon. The consumer has the ability to change his environment, e.g., from at home to a friend. The mobile phone discovers the other P2 service installed in the friend's home to exchange information. Is content accessed in another domain (e.g. YouTube), the other service acts as data consumer and personalizes the content according the P2 profile. Thus, e.g., on YouTube music videos according the taste of the consumer shall be presented. Figure 12 illustrates the basic system architecture and Fig. 13 the screenshots of the three different P2 applications.

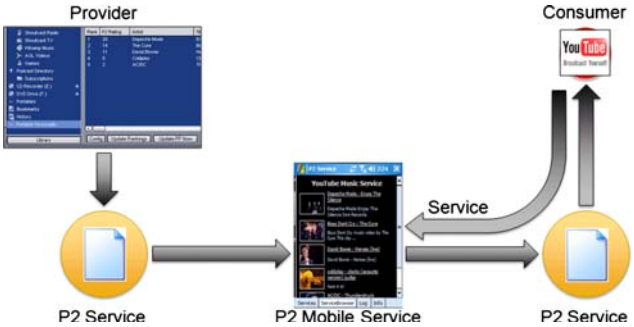


**Fig. 12** Overview of the P2 system architecture with multiple provider and consumer nodes interconnected through the distributive P2 daemon network


**Fig. 13** Screenshot of the different P2 components in use for the audio-visual content scenario, where the profile is exchanged via Bluetooth

## 6 Discussion and evaluation

To evaluate P2, we have to discuss different aspects of the system: system architecture level, metadata merging, and mining algorithms, and further development of providers and consumers.

On architecture level, P2 provides a slim architecture to comply with today's trend towards increased mobility and embedded systems. The burden to implement plug-ins for 3rd party applications is rather small. This is due to the fact, that a generalized API exists, which wraps functionality for different metadata provider. P2 also relies on an opened architecture for metadata handling, and can cope with a wide variety of metadata standards, which is the key-strength of the platform. Simple annotations and adding rules for merging of metadata are sufficient to become compliant with P2.

P2 is currently released in its first version and contains a stable package for its AmbiNET component, various P2 Provider in form of plug-ins (e.g. Skype, Microsoft messenger, WinAmp, YouTube), and an intermediate release of the P2 Service. Currently the system is capable of managing FOAF metadata and audio-visual metadata. AmbiNET is implemented in C# as well as in Java. The application-specific plug-ins are implemented either in C#, C++, or Symbian.

The current development team works towards an extension towards services provided from lastFM. However, as P2 is an evolving open-source platform, many different plug-ins and extensions will be provided in the near future. The latest version of P2 are made available to a selected community in 2009 (www.portable-personality.org). An earlier version can be found on Sourceforge.

The development of metadata merging and mining algorithms is one of the main fields of investigation within the P2 projects. The goal is to find an adequate algorithm, especially focusing on context metadata attributes. As the main aim during recent years was the development of a test-environment, currently we are investigating rough sets, fuzzy logic, or Bayesian networks to mine the consumer data. However, as the main aim is the provision of a personality profile, this still will require time. To provide a computer-readable form of 'personality' is still a major challenge and requires much research work across domains. However, we hope to extend the framework onto this level within the next years to come, and a first evaluation of algorithms has been performed in [30]. Currently, synchronization and versioning is based on the annotation of metadata with P2 tags. At this stage, P2 nicely performs on annotated data. However, synchronization and versioning on higher level algorithms have to be considered in the future to allow system scalability. Algorithms fitting to the case of P2 are currently under investigation.

The development of additional provider and consumers is rapidly evolving, as it is straight forward engineering and implementation work. We are investigating web services such as IMDB, lastFM, or Amazon among many others.

In short, P2 is open-source and allows the implementation of personalized services in an ubiquitous environment. The basic architecture has been developed, and is currently under heavy extension. A release of version 0.9 of the platform to the general public is planned in 2009, and will include various P2 provider, P2 clients, and a stable release of the P2 service for mobile and stationary domains. Currently this version is under field testing.

**Acknowledgments** This work was supported by the Academy of Finland, project No. 213462 (Finnish Centre of Excellence Program (2006–2011).

## Appendix A: Metadata coming from various p2 components and their resulting merging

**Table 5** Metadata coming from various P2 components and their resulting merging (personal data has been made anonymous to warrant privacy)

**DEFINITION OF THE PERSON USING SKYPE (NAME, NICKNAME, EMAIL, SKYPE ID) AND THE METADATA MERGEMODES FOR THIS SPECIFIC ENTRY:**

```
<foaf:Person p2:mergekeys="foaf:mbox" p2:mergemodes="foaf:Person,IFDIFFERENT_INSERTAFTER_MERGE">
  <foaf:name p2:mergemodes="*foaf:name,IFDIFFERENT_INSERTAFTER_TAKEOLDER"
p2:sourcekeys="*foaf:name,1">Artur Lugmayr</foaf:name>
  <foaf:nick p2:mergemodes="*foaf:nick,IFDIFFERENT_INSERTAFTER_TAKEOLDER"
p2:sourcekeys="*foaf:nick,1">lartur</foaf:nick>
  <foaf:mbox p2:mergekeys="*rdf:resource" rdf:resource="lartur@acm.org" p2:sourcekeys="*rdf:resource,1"/>
  <sn:skypeChatID p2:mergemodes="sn:skypeChatID,IFDIFFERENT_INSERTAFTER_TAKEOLDER"
p2:sourcekeys="*sn:skypeChatID,1">lartur</sn:skypeChatID>
```

**PERSONS KNOWN BY THE PERSON CURRENTLY USING SKYPE INCLUDING THE FRIEND'S EMAIL AND NAME AND THE METADATAMERGEMODES FOR THIS SPECIFIC ENTRY:**

```
  <foaf:knows p2:mergemodes="foaf:knows,IFEQUAL_MERGE_TAKEOLDER">
    <foaf:Person p2:mergekeys="*sn:skypeChatID"
p2:mergemodes="foaf:Person,IFDIFFERENT_INSERTAFTER_MERGE">
      <foaf:name p2:mergemodes="*foaf:name,IFDIFFERENT_TAKENEWER_TAKEOLDER"
p2:sourcekeys="*foaf:name,1">Susi Sorglos</foaf:name>
      <foaf:nick p2:mergemodes="*foaf:nick,IFDIFFERENT_TAKENEWER_TAKEOLDER"
p2:sourcekeys="*foaf:nick,1">susi</foaf:nick>
      <foaf:mbox p2:mergemodes="*rdf:resource,IFDIFFERENT_TAKENEWER_TAKEOLDER"
rdf:resource="susi.sorglos@sorglos.de" p2:sourcekeys="*rdf:resource,1"/>
      <sn:skypeChatID p2:sourcekeys="*sn:skypeChatID,1">susi.sorglos</sn:skypeChatID>
    </foaf:Person>
  </foaf:knows>
 </foaf:Person>
</rdf:RDF>
...
```

**Table 5** continued

**DEFINITION OF THE PERSON USING MICROSOFT MESSANGER (NAME, NICKNAME, EMAIL, SKYPE ID) AND THE METADATA MERGEMODES FOR THIS SPECIFIC ENTRY:**

```
<foaf:Person p2:mergekeys="foaf:mbox" p2:mergemodes="foaf:Person,IFDIFFERENT_INSERTAFTER_MERGE">
  <foaf:name p2:mergemodes="*foaf:name,IFDIFFERENT_INSERTAFTER_TAKEOLDER"
p2:sourcekeys="*foaf:name,2">Artur Lugmayr</foaf:name>
  <foaf:nick p2:mergemodes="*foaf:nick,IFDIFFERENT_INSERTAFTER_TAKEOLDER"
p2:sourcekeys="*foaf:nick,2">lartur</foaf:nick>
  <foaf:depiction p2:mergemodes="*rdf:resource,IFDIFFERENT_INSERTAFTER_TAKEOLDER"
rdf:resource="url_to_resource" p2:sourcekeys="*rdf:resource,2"/>
  <foaf:mbox p2:mergekeys="*rdf:resource" rdf:resource="lartur@acm.org" p2:sourcekeys="*rdf:resource,2"/>
```

**PERSONS KNOWN BY THE PERSON CURRENTLY USING MICROSOFT MESSANGER INCLUDING THE FRIEND'S EMAIL AND NAME AND THE METADATAMERGEMODES FOR THIS SPECIFIC ENTRY:**

```
  <foaf:knows p2:mergemodes="foaf:knows,IFEQUAL_MERGE_TAKEOLDER">
    <foaf:Person p2:mergekeys="*foaf:msnChatID"
p2:mergemodes="foaf:Person,IFDIFFERENT_INSERTAFTER_MERGE">
      <foaf:name p2:mergemodes="*foaf:name,IFDIFFERENT_TAKENEWER_TAKEOLDER"
p2:sourcekeys="*foaf:name,2">Susi Sorglos</foaf:name>
      <foaf:nick p2:mergemodes="*foaf:nick,IFDIFFERENT_TAKENEWER_TAKEOLDER"
p2:sourcekeys="*foaf:nick,2">Sushi</foaf:nick>
      <foaf:mbox p2:mergemodes="*rdf:resource,IFDIFFERENT_TAKENEWER_TAKEOLDER"
rdf:resource="sushi@hotmail.com" p2:sourcekeys="*rdf:resource,2"/>
      <foaf:msnChatID p2:sourcekeys="*foaf:msnChatID,2">sushi@hotmail.com</foaf:msnChatID>
    </foaf:Person>
  </foaf:knows>
 </foaf:Person>
</rdf:RDF>
```

**EXAMPLE FOR MERGED METADATA PROFILES FROM SKYPE AND MICROSOFT MESSANGER:**

```
...
<foaf:knows p2:mergemodes="foaf:knows,IFEQUAL_MERGE_TAKEOLDER">
<foaf:Person p2:mergekeys="*sn:skypeChatID" p2:mergemodes="foaf:Person,IFDIFFERENT_INSERTAFTER_MERGE">
      <foaf:name p2:mergemodes="*foaf:name,IFDIFFERENT_INSERTAFTER_TAKEOLDER"
p2:sourcekeys="*foaf:name,1">{Susi Sorglos}</foaf:name>
      <sn:skypeChatID p2:sourcekeys="*sn:skypeChatID,1">{susi.sorglos}</sn:skypeChatID>
    </foaf:Person>
...
    <foaf:Person p2:mergekeys="*foaf:msnChatID"
p2:mergemodes="foaf:Person,IFDIFFERENT_INSERTAFTER_MERGE">
      <foaf:name p2:mergemodes="*foaf:name,IFDIFFERENT_TAKENEWER_TAKEOLDER"
p2:sourcekeys="*foaf:name,2">{Susi Sorglos}</foaf:name>
      <foaf:nick p2:mergemodes="*foaf:nick,IFDIFFERENT_TAKENEWER_TAKEOLDER"
p2:sourcekeys="*foaf:nick,2">{Sushi}</foaf:nick>
      <foaf:mbox p2:mergemodes="*rdf:resource,IFDIFFERENT_TAKENEWER_TAKEOLDER"
rdf:resource="{sushi@hotmail.com}" p2:sourcekeys="*rdf:resource,2" />
      <foaf:msnChatID p2:sourcekeys="*foaf:msnChatID,2">{sushi@hotmail.com}</foaf:msnChatID>
    </foaf:Person>
  </foaf:knows>
```

## References


1. Adobe Extensible Metadata Platform (XMP): http://www.adobe.com/products/xmp/main.html
2. Ambady, N., Rosenthal, R.: Thin slices of expressive behaviour as predictors of interpersonal consequences: a meta-analysis. Psychol. Bull. **111**, 256–274 (1992). doi:10.1037/0033-2909.111.2.256
3. Boda, P., Chande, S., Gupta, N., Hartikainen, E., Autere, S.: Flexibility and efficiency through personalisation? Experiments with a conversational Program Guide Information System. In: EACL-03 Workshop on "Dialogue Systems: interaction, adaptation and styles of management". Budapest, Hungary (2003)
4. Bruns, V., Reymann, S., Lugmayr, A.: Distributed profiling in a partitioned ambient network. In: Proceedings of the 6th international conference on Mobile and ubiquitous multimedia Oulu. ACM, Finland (2007)
5. DCMI: Dublin Core Metadata Initiative (DCMI). http://dublincore.org/
6. Description of a Career (DOAC). http://ramonantonio.net/doac/
7. Ehrmantraut, M., Haerder, T., Witting, H., Steinmetz, R.: The personal electronic program guide—towards the pre-selection of individual TV programs. In: ACM (1996)
8. Hossain, M.A., Atrey, P.K., El Saddik, A.: Management of ambient media preferences in distributed environments for service personalization. In: International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 2008) (2008)
9. ID3: http://www.id3.org/id3v2-00
10. International Press Telecommunications Council (IPTC): http://www.iptc.org/IPTC4XMP/
11. ISTAG: Scenarios for Ambient Intelligence in 2010—Final Report (Feb. 2001), http://www.cordis.lu/ist/istag.htm (2001)
12. ISTAG.: Ambient intelligence: from vision to reality. European Union: IST Advisory Group, draft report (2003)
13. Lehikoinen, J., Aaltonen, A., Huuskonen, P., Salminen, I.: Personal content experience: managing digital life in the mobile age. Wiley-Interscience, New York (2007)
14. Leslie, G.S.: Why and how CARPE should be personal? In: Proceedings of the 2nd ACM workshop on Continuous archival and retrieval of personal experiences Hilton. ACM Press, Singapore (2005)
15. Liberty Alliance: http://www.projectliberty.org/
16. Mackiewicz, J., Moeller, R.: Why people perceive typefaces to have different personalities, p. 304 (2004)
17. Mateas, M., Stern, A.: A behavior language for story-based believable agents. Intell. Syst. IEEE **17**, 39 (2002). see also IEEE Intelligent Systems and Their Applications. doi:10.1109/MIS.2002.1024751
18. McCluskey, E., Mahmood, A., Lu, D.: Concurrent fault detection using a watchdog prozessor and assertions. In: Proceedings of the IEEE International Test Conference Philadelphia (1983)
19. Open, I.D.: http://openid.net/
20. Ören, T.I., Ghasem-Aghaee, N.: Personality representation processable in fuzzy logic for human behavior simulation. In: Summer Computer Simulation Conference, pp. 11–18. Montreal, PQ, Canada (2003)
21. Reis, H.T., Collins, W.A., Berscheid, E.: The relationship context of human behavior and development. Psychol. Bull. **126**, 844–872 (2000). <Go to ISI>://000165365200004
22. Reymann, S., Kemper, S.: AMBINET—a lightweight development framework for ambient application design. In: NAMU Lab. vol. MSc. Tampere: Tampere University of Technology (TUT) (2008)
23. Reymann, S., Bruns, V., Lugmayr, A.: P2—Portable personality a middleware solution for smart user profile management and distribution. In: Interactive TV: A Shared Experience, TISCP Adjunct Proceedings of EuroITV 2007 Amsterdam, (2007)
24. Reymann, S., Kemper, S., Lugmayr, A.: AmbiNet—a lightweight open source development framework for personalized ambient entertainment. In: Lugmayr, A., Kemper, S., Obrist, M., Mirlacher, T., Tscheligi, M. (eds.) Changing Television Environments, TICSP Adjunct Proceedings of EuroITV, vol. 42. Tampere University of Technology (TUT), Salzburg, Austria (2008)
25. Riva, G., Vatalaro, F., Davide, F., Alcaniz, M.: Ambient Intelligence. IOS Press (2005). http://ambientintelligence.org
26. SchemaWeb: http://www.schemaweb.info/default.aspx
27. SyncML, Building an Industy-Wide Mobile Data Syncronization Protocol: Version 1.0
28. The Friend of a Friend (FOAF) Project: http://www.foaf-project.org/
29. Tseng, B.L., Lin, C.-Y., Smith, J.R.: Using MPEG-7 and MPEG-21 for personalizing video. Multimedia IEEE **11**, 42–52 (2004). doi:10.1109/MMUL.2004.1261105
30. Uhlmann, S., Lugmayr, A.: Personalization algorithms for portable personality. In: Proceedings of the 12th international conference on Entertainment and media in the ubiquitous era Tampere. ACM, Finland (2008)
31. W3C: World Wide Web Consortium (W3C). www.w3c.org
32. W3C Multimedia Semantics Incubator Group. http://www.w3.org/2005/Incubator/mmsem/
33. World Wide Web Consortium (W3C): XML Schema Part 2: Datatypes (2001)
34. World Wide Web Consortium (W3C): XML Schema Part 1: Structures (2001)
35. World Wide Web Consortium (W3C): XML Schema Part 0: Primer (2001)
36. WWI Ambient Networks Project 507134.: D.1.5 AN Framework Architecture—Mobile and Wireless Systems beyond 3G. 6th Framework Programme