We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

5

An AmI-enabled OSGi platform based on socio-semantic technologies*

Ana Fernández Vilas, Rebeca P. Díaz Redondo, José J. Pazos Arias,
Manuel Ramos Cabrer, Alberto Gil Solla and Jorge García Duque
Department of Telematics Engineering, University of Vigo
Spain

Abstract
Ideally, smart homes should make its inhabitants' lives more comfortable by anticipating their needs and satisfying their preferences. With this aim, we introduce an approach for context-aware personalized smart homes based on three pillars: the Open Service Gateway Initiative (OSGi) platform, the Semantic Web philosophy and the collaborative tagging trends. By combining these fields, we enrich the OSGi service-oriented architecture by providing a semantical conceptualization of services at home. On the top of this semantic formalization of services, we support context-awareness personalization by using a dynamic and non previously agreed structure for modeling context: a folksonomy. This socio-semantic approach to the problem takes into account the heterogeneous nature of the devices which provide contextual information so that defining a previously agreed constrained vocabulary for context is unrealistic.

1. Introduction

To realize the vision of Ambient Intelligence (AmI), smart homes should make its inhabitants' lives more comfortable by anticipating their needs and satisfying their preferences, that is, smart homes should be able to automatically select and provide the adequate services to its inhabitants' at the right time. This personalization should be done accordingly to the preferences and behavior of the inhabitants and their surrounding environment. Obviously, this goal entails joining efforts coming from different research fields like residential gateways, context-awareness, personalization, sensors, home networks, etc. With this aim, we introduce an approach for context-aware personalized smart homes based on two main pillars: the Open Service Gateway Initiative (OSGi (*OSGi Service Platform, Core Specification, Release 4*, 2005)) platform and the Semantic Web philosophy. By combining both fields, we enrich the OSGi service-oriented architecture by providing a semantical conceptualization of (i) services at home, (ii) contextual information and (iii) inhabitants' preferences. This ontological structure supports reasoning about the captured behavior and inferring new knowledge.

*Work funded by the Ministerio de Educación y Ciencia (Gobierno de España) research project TSI2007-61599, by the Consellería de Educación e Ordenación Universitaria (Xunta de Galicia) incentives file 2007/000016-0, and by the Programa de Promoción Xeral da Investigación de la Consellería de Innovación, Industria e Comercio (Xunta de Galicia) PGIDIT05PXIC32204PN

OSGi specification is currently the most widely adopted technology for building control systems for the networked home. However, its mechanism for service discovering (based on syntactic match making) and invocation assumes the client to know both the name of the service interface and its method's signatures, respectively. Because of this, in previous work we have defined a Semantic OSGi platform (Díaz Redondo et al., 2008) inspired by the Semantic Web conception (Berners-Lee et al., 2001), which is based on the markup of OSGi services by using appropriate ontologies. In the Semantic OSGi platform we have defined, OSGi services describe their properties and capabilities so that any other software element can automatically determine its purpose (semantic discovery) and how to invoke them.

The work introduced in this paper means another step to enrich the Semantic OSGi platform to support both context-awareness and personalization. We pursue a platform which is able to learn about the preferred services for inhabitants in specific contexts and, consequently, to invoke those appropriate services when applicable. This entails the services must be automatically discovered, selected according to both the contextual information and user preferences and launched using the most adequate invocation parameters for the environmental atmosphere at home. In order to do that, we postulate that a previously agreed vocabulary for modelling context does not respond to the dynamic, heterogeneous and distributed nature which is intrinsic to the sources of contextual information. In such an environment, the top-down strategy in ontologies, which remains valid for services categorization, turns into inappropriate. A solution which establishes the context model from the bottom (from the devices, sensors, services or even inhabitants at home in an unconstrained way) can make context management simple, flexible, interoperable and maintenable.

To introduce the context-aware personalized extension to OSGi we propose, the paper is organized as follows. Sect. 2 and Sect. 3 overview the characteristics of the current OSGi framework and the Semantic OSGi platform we have previously defined (Díaz Redondo et al., 2008). Then, Sect. 4 introduce the notion of context and preference in out knowledge base. The details related to our proposal of adding context-awareness and personalization to the Semantic OSGi platform is described in Sect. 5 and Sect. 6. Once the approach has been introduced, the details about the implementation of the prototype are presented in Sect. 7. Finally, Sect. 8 is devoted to analyze other approaches related with the smart home technologies. Finally, a brief discussion including conclusions and future work is presented in Sect. 9, respectively.

2. Overviewing the Semantic OSGi Platform

The OSGi platform consists of a Java Virtual Machine (JVM), a set of running components called *bundles*, and an OSGi framework (*OSGi Service Platform, Core Specification, Release 4*, 2005). A bundle is a Java archive (JAR) and the minimal deliverable application in OSGi, whereas the minimal unit of functionality is a *service*. An OSGi service is defined by its Service Interface, specifying the service's public methods, and is implemented as a Service Object, owned by, and runs within, a bundle. So, a bundle is designed as a set of cooperating services, which any application might discover after the services are published in the OSGi Service Registry.

OSGi aims to provide a flexible platform. However, service discovery and use assume the client has a great deal of information about the service it wants to use. On the one hand, since service discovery is based on syntactic match making, the client must know the service's interface name or at least the keywords in its properties. This mechanism has several drawbacks, including problems with synonyms (semantically similar but syntactically different services) and homonyms (syntactically equivalent but semantically different services). On the other hand, for invocation purposes the client must know the service's method signatures. This is clearly an obstacle in a pervasive environment because it prevents bundles without prior knowledge of the service interface from dynamically invoking the service.

These requirements conflict with the remote management of OSGi applications, one of the platform's mainstays. How can OSGi applications be remotely deployed and work properly on every customer platform if home devices can differ from one customer to another? What is more, as the OSGi specification notes, customers can use the home service platform to run services from different service providers. So, it is hardly realistic to suppose that a specific provider knows, a priori, the interfaces that match other providers' services.

In this context, OSGi's simple syntactic name matching is not enough. So, we proposed a semantic approach to service discovery that turns OSGi into a Semantic OSGi platform (Díaz Redondo et al., 2008). Our approach was based on the markup of OSGi services by using appropriate ontologies, since they support a common understanding for both the service requester and the service provider. More precisely, the table-based structure in the OSGi Service Registry was replaced by an ontological structure integrating: semantic classification, semantic description of properties and information of invocation. Additionally, we have also provided a way for OSGi bundles to register semantically their services and to get the correct references to the needed ones. The work in this proposal is summarized in the following sections.

2.1 The OWL-OS Framework

The main contribution of the OWL-OS framework is re-structuring the table approach in OSGi Registry into an ontological structure which conceptualizes the main aspects of the smart home, that is, its resources and its services. The former can be conceptualized by any structure; OWL-OS is agnostic at this respect. The latter relies on OWL-OSGi Services ontology (OWL-OS) in Fig. 1. OWL-OS semantically describe OSGi services as an extension of OWL-S (*OWL-S: Semantic Markup for Web Services. 1.1 Release*, 2004), and OWL-based Web Services Ontology which provides a semantic markup language specially thought to specify Web Services in unambiguous and computer-interpretable form.

2.2 The OWL-OS Ontology

The OWL-S ontology is organized to provide three essential types of knowledge about any `Service` provided by a `Resource` (upper part of Fig. 1): (i) the `ServiceProfile` tells "*what the service does*"; (ii) the `ServiceModel` explains "*how the service works*"; and (iii) the

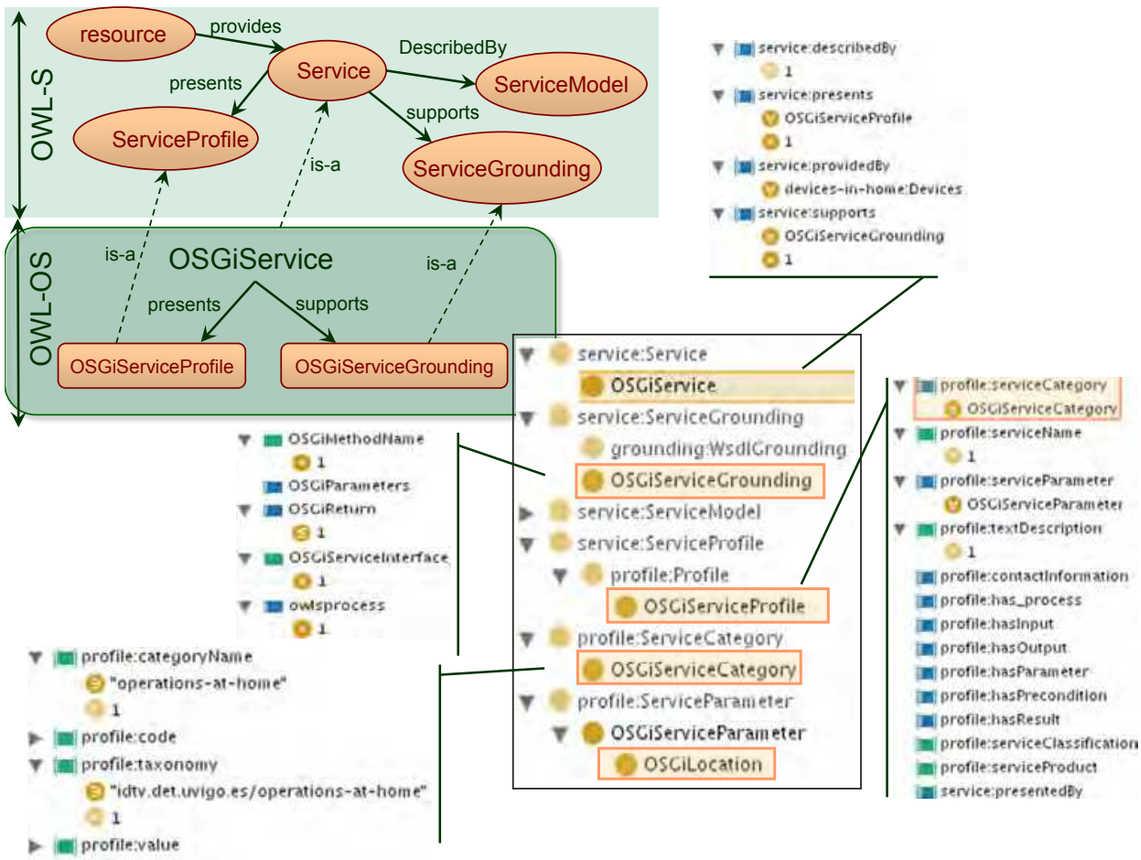


Fig. 1. The *OWL-OS ontology* (OWL-OSGi Services Ontology)

`ServiceGrouding` details "*how to access it*". We have adapted this ontology to the peculiarities of the OSGi services by defining the OWL-OS ontology (*OWL-OSGi Services* ontology).

So, an *OSGiService* is characterized by: (i) its *OSGiServiceProfile*; (ii) its *OSGiServiceGrounding*; and (iii) some instance in the ontology which supports smart home modelling[1]. Finally, we do not adopt any restriction regarding the *ServiceModel* (how the service works) because the OWL-S one fits perfectly with the OSGi perspective.

Inspired in the OWL-S ServiceProfile, the **OSGiServiceProfile** (subclass) describes an OSGi service by using two main fields (see Fig. 1): (i) the *OSGiServiceCategory* classifies the service in an ontology representing the typical services at the smart home (`operations-at-home` in our case[2]); and (ii) the *OSGiServiceParameter*, an expandable list of properties that may help the requester to find the service. Note that any OSGiServiceProfile may provide more than one categorical description; for instance, the service "*opening a window*" can be both an

[1] As it is said before, the solution is agnostic with respect to the conceptualization of smart home domain. Anyway, we use `devices-in-home` ontology for implementation

[2] Once again the solution is agnostic with respect to this ontology.

airing service or a lighting service. Any other information available about the OSGi service is maintained as a list of OSGiServiceParameters. Finally, the **OSGiServiceGrounding** (subclass of the ServiceGrounding) provides a suitable way for mapping the OSGiServiceProfile to the functionality offered, through a Service Interface, by a bundle running in the OSGi framework.

3. Operation in OWL-OS Framework

The operation in the semantic Framework imitate the way of doing in an OSGi platform. An OSGi service is defined by a *Service Interface*, specifying the service's public methods, and is implemented as a *Service Object*, owned by, and runs within, a bundle. The bundle is in charge of registering the service object with the OSGi Service Registry so that its functionality is available to other bundles. For instance, in Fig. 2 bundle A registers the service object (`serviceobj`) jointly with a dictionary object (`props`), under the `Printing` Service Interface as follows:

```
registerService("Printing", serviceobj, props)
```

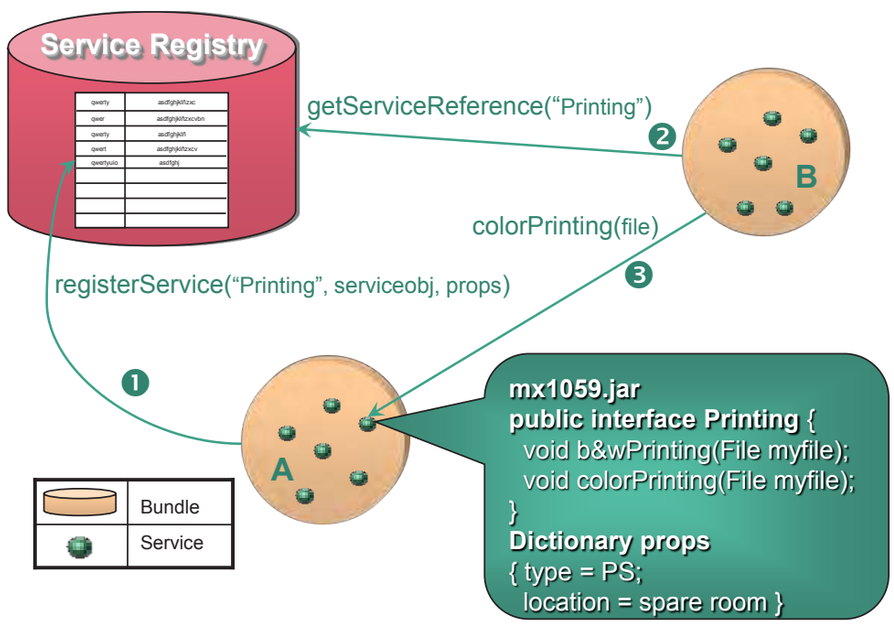


Fig. 2. Operation in the OSGi Platform

Secondly, registered services are referenced through *Service Reference* objects, which maintain the properties and other meta information about the service. Bundles can obtain an OSGi service, actively by syntactic search (`getServiceReference()`), or passively by the event mechanism provided by the framework (*Service Listeners*). For instance, bundle *B* can obtain the Service Reference object corresponding to the service registered by bundle *A* as follows:

```
ref = getServiceReference("Printing")
```

Once bundle *B* has obtained the Service Reference object, it must obtain the service object itself to be able to use the required method, `colorPrinting` in this case:

```
printer = getService(ref)
printer.colorPrinting(file)
```

On another hand, the OSGi framework also offers the possibility of refining the search by using the `getServiceReferences()` method, whose input parameters are (i) the name of the Service Interface and (ii) filtering information to help the matching process. The filter syntax is based on the LDAP language (*Lightweight Directory Access Protocol*) (Howes, 1996). For instance, to obtain the list of all PostScript services belonging to the Service Interface `Printing`, a bundle could use the command:

```
ref=getServiceReferences("Printing", "(type=PS)")
```

where the filter `type=PS` is matched against the dictionary of properties provided on registration.

3.1 Operating semantically

The OWL-OS framework redraw the OSGi operation (registering, discovering and invoking OSGi services) in the following way. For **registering purposes**, we propose the bundle developer specifies the semantic information in the bundle's manifest file using a set of new headers: the *OWL-OS headers*. According to this information, the Semantic OSGi Service Registry may create, if necessary, new individuals and/or classes when a new service is registered in the ontology. So, OSGi bundles use the same registering method (`registerService()`) of the OSGi specification (see Fig. 3). Note we maintain the dictionary of properties in the registering method just for compatibility reasons with the standard Service Registry. The OWL-OS parameters of service, like `OSGiLocation` or other service-dependent parameters (`printFormat` in our example) are assumed to be managed inside the OWL-OS ontology. Their initial values, if any, are extracted from the manifest file (for instance the value `#PS` for `printFormat`).
For the semantic **discovery mechanism** we have proposed to add a method inspired in the OSGi original one:

```
semGetServiceReference(String OntURI, String Cat)
```

Thus, instead of providing the name of the Service Interface, the requesting bundle specifies the name of the service category (`Cat`) and which ontology is using to classify them (`OntURI`). Finally, to **invoke** semantic services, we have added get-set style methods to the *ServiceReference* interface of the OSGi Specification. So, the requester bundle can discover the name of the public method and the parameters in order to construct the invoking primitive.

4. Modeling context and preferences

To establish a context-aware and personalized solution on top of our OWL-OS framework, we need an automated model of both the context and the preferences of the user. These two models are not independent each other so that the preferences strongly depend on the context of the user (where, when, etc.).

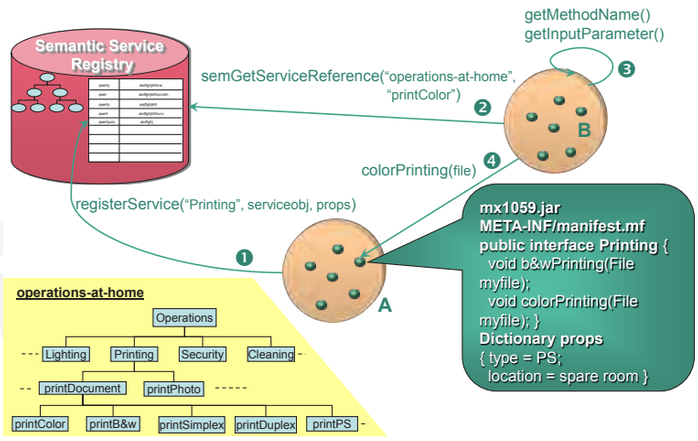


Fig. 3. Operation in the Semantic OSGi Platform

4.1 Context modeling

For context modeling, we assume the Zimmermann (Zimmermann et al., 2005) vision of **CXMS (ConteXt Management System)** as a system which involves the construction, integration, and administration of context-aware behavior. Our proposal assumes an external CXMS that is integrated in the OSGi Framework as one more OSGi service. The key question in the smart home is what structures we use to model the context.

A user's context can be defined broadly as the circumstances or situations in which a computing task takes place (Meyer & Rakotonirainy, 2003). More in general, context includes any information that can be used to characterize the situation of an entity (person, physical or computational object) in any domain (home domain, health domain, etc.). In the smart home domain, the home is plenty of sensors which determine various types of contexts of its inhabitants (location, mood, etc.) while applications in the home use this information to provide context-aware services. The great variety of that information makes the task of formalizing a context model specially complex.

Several works in the literature have tackled with the problem of constructing an ontology for contextual information (see Sect. 8). According to the Gruber's popular definition, an ontology is a description of the concepts and relationships that can exist for an agent or a community of agents. These agents (who offer software/data/services) identify and specify some common conceptualization of the data so that systems can be built which interoperate on those specifications. We believe constructing this conceptualization in a up-bottom and static way is never suitable in a domain which is inherently heterogeneous an even subjective. Instead of formalizing this knowledge by using an ontology, we propose using a folksonomy to model context. The term folksonomy was coined by Thomas Vander Wal (Wal, 2007) as 'the user-created bottom-up categorical structure development with an emergent thesaurus'. This

emergent structure comes up from the sensor's and device's metadata in a natural way. Moreover, a folksonomy-based approach to context modeling can aggregate not only real data and metadata from devices at home but even user-provided information about his context. Similar approach has been considered in (Mizzaro et al., 2009) in the field of information retrieval.

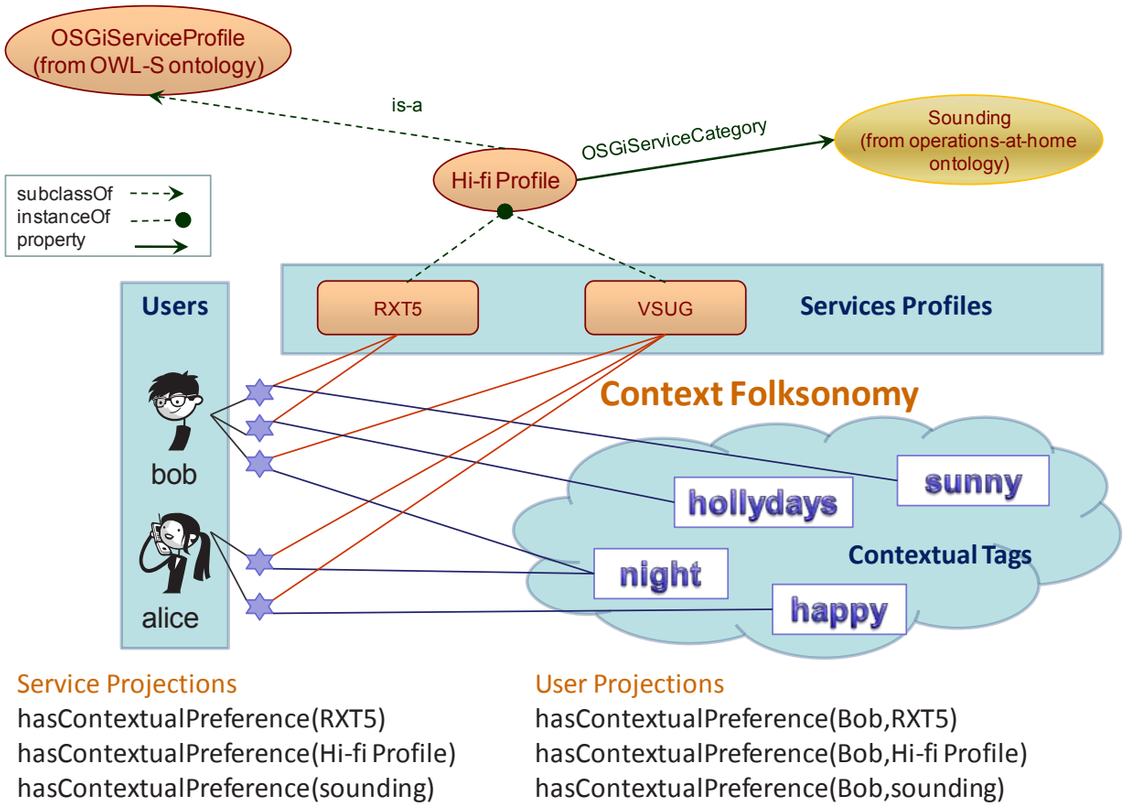


Fig. 4. A folksonomy-based CSMX

In our approach, the actual context of the smart home is represented by the set of tags obtained from the devices and even the inhabitants at home; these tags range over an uncontrolled vocabulary. Beyond the actual context, the information about the contexts in which inhabitants use the services at home is modeled by means of the folksonomic-structure in figure 4. In this figure, we can observe the three elements which are involved in establishing the contextual characterization of a service from the point of view of a specific user (users, service profiles and contextual tags). Mathematically speaking, this folksonomic structure can be described (similar to the characterization in (Hotho, Jaschke, Schmitz & Stumme, 2006b)) as a tuple $F_C := (U, T, S, Y)$ where $U = \{u_1, \cdots, u_m\}$, $T = \{t_1, \cdots, t_l\}$, and $S = \{s_1, \cdots, s_k\}$, are finite sets of items (services in our case), tags (contextual tags in our case) and users. $Y \subseteq UxTxS$ is a ternary relation whose elements are called tag assignments. In our contextual folksonomy for services, tags assignments are the tags which describe the context where an inhabitant uses a service at home. As it is shown in the following section, according to this structure the contextual characterization of a service is represented by a set of tags (graphically by a tag cloud) with their respective weights, which are proportional to the number of users who

have used this service in a context described with this tag. Similarly, tags are linked in a way that the more times the tags are assigned together to the same context or situation, the highest the weight of their relationship. This structure is then used to establish relationships between contexts in our semantic OSGi platform.

More than the universe of tag assignments to context, for personalization purposes we want to know the contexts of a specific user. From the tripartite graph above, we can obtain the bipartite graphs of personal folksonomies (personomies), which represents the contextual information of a single inhabitant u using the services at home. Mathematically speaking, the context personomy P_C^u of a user u is the restriction of the folksonomy F_C to u, that is the set of user's tags which are assigned to the set of user's services.

4.2 Preference Model

Until now, we have defined a model to represent the contexts in which users invocate services at home. However, a personalized context-aware solution requires to establish a preference model, also connected with the CXMS, to adapt the context-aware response of the smart home to the user's characteristics. With this aim, we define a **preference model** that stores information about any entity s in the service model (class or instance) from the point of view of any user u in the system, concretely, the context of application C (time, place, etc.) and the preference (an index DOP (*Degree of Preference*)) about the invocation of this service by this user. The value of the index $DOP(u,s)$ reflects how much the user u is interested in using the service s; and the element $C(u,s)$ reflects the characteristics of the past contexts where the user has accessed to the service s.

Based on the above conception, our preference model (Fig. 5) enriches the OWL-OS ontology with the property `hasContextualPreference`; a complex property which consists of two subproperties: `hasContext` and `hasDOP`. The ontological preference model of the inhabitant u is a projection of this ontology where each entity s (instance or type, say class, of service) is characterized by the contextual preference of the specific user, that is, the context $C(s,u)$ where the service has been used by this user, and the index of preference $DOP(s,u)$ the user about this service. Both properties are defined on top on the ontology so that the pair $\{DOP, C\}$ qualifies every class or instance in the ontology. The value of $DOP(s,u)$ and $C(s,u)$ for a specific entity s in the ontological preference model is recomputed whenever the inhabitant u uses the service by (1) weighting DOP with the number of service instances in the same category and (2) aggregating in C the new contextual information (as it is shown in the following section). From instance-scope DOPs and Cs (`RXT5` and `VSUG` in figure 4) the ones corresponding to classes (`Hi-fi Profile` in figure 4) are obtained by recursively propagating DOPs and Cs up through the hierarchy defined by *operations at home*.

The preference model allows to vary the quantity of collective intelligence to be taken into account when personalizing service selection in a context aware way. At the maximal level of collective intelligence, we have a projection of the ontology over the contextual folksonomy. In that case the `hasContextualPreference` property ranges over the contextual folksonomy F_C and the DOP is computed for all the users considered in the system. At the minimal level

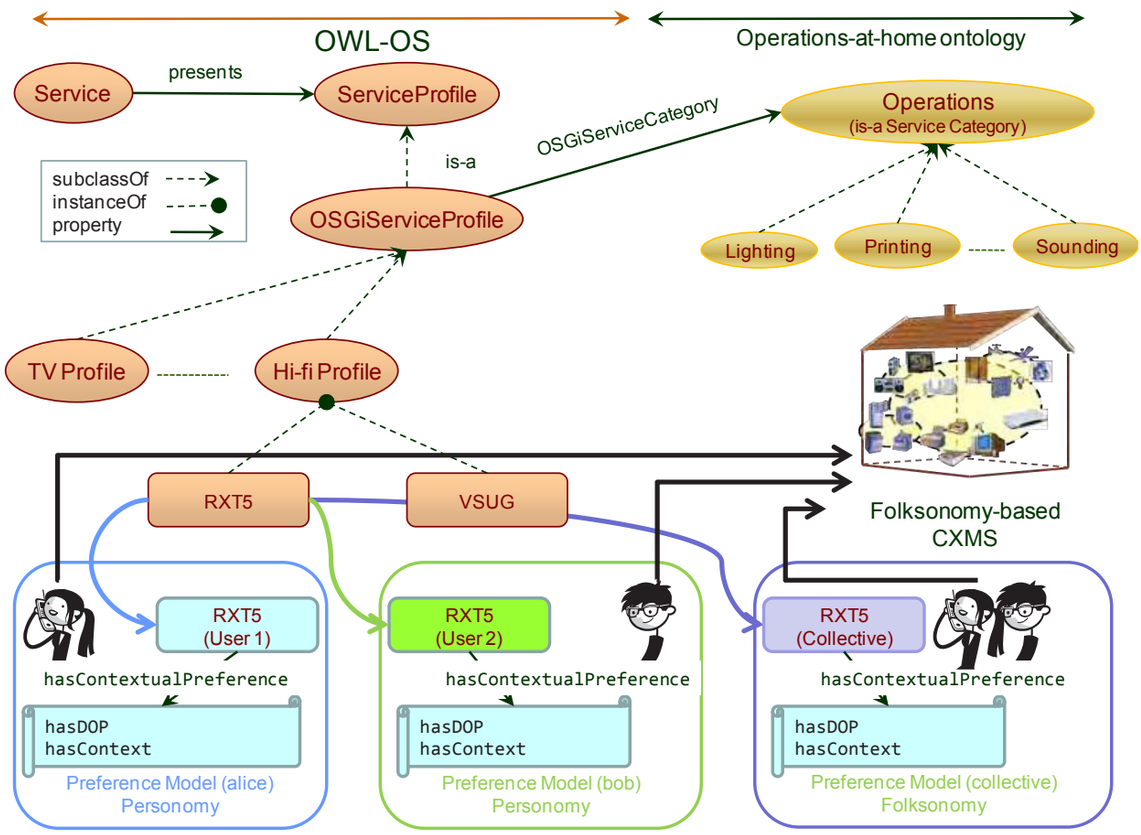


Fig. 5. Context-aware Preference Model for the services at home

of collective intelligence, we have a projection of the ontology over a contextual personomy. In that case the `hasContextualPreference` property ranges over the contextual folksonomy F_C^u and the *DOP* is computed only for the specific user u.

5. Folksonomic support to Context-aware Personalization

5.1 Describing the context of a service

A service in the smart home is contextually described by a tag cloud. From the definition of the folksonomy $F_C := (U, T, S, Y)$, a tag cloud for a specific service profile instance s (say service), is defined as $TC_S(s) = \{t, w(s,t)\}$ such that $(u,t,s) \in Y$ and the weight is computed as follows. For a service s, the weight assigned to a contextual tag t corresponds to the occurrence frequency of the tag in contexts for the service s. We normalize the weights so that the sum of the weights assigned to the tags in the tag cloud is equal to 1.

Being $T(s) = \{t_1, t_2, \ldots, t_{n_i}\}$ the set of tags which have been used to describe the usage context of the service s and $m : T(s) \to \mathbb{N}$ the function that relates each tag with its multiplicity in the multiset, i.e., the number of times that the tag t have been used to describe the context of the service s, denoted as $m(t, T(s))$. We define the function $w(s,t) : T(s) \to \mathbb{R}$ as the one which relates each tag $t \in T(s)$ with its weight in the multiset which is calculated as follows:

$$w(s,t) = \frac{m(t, T(s))}{|T(s)|} \tag{1}$$

where $|T(s)|$ is the cardinality of the multiset, i.e., the result of adding the multiplicities of all of its tags:

$$|T(s)| = \sum_{\forall k / t_k \in T(s)} m(t_k, T(s))$$

Hence,

$$\sum_{\forall k / t_k \in T(s)} w(s,t_k) = 1 \tag{2}$$

In this approach, apart from storing the tag cloud for the contextual information of a service, it is also important to know another important parameter, the Index Of Popularity (IOP), calculated from the number of contextual tags assigned to this service with respect to the total number of contextual tags assigned in the system:

$$IOP(s) = \frac{|T(s)|}{\sum\limits_{\forall k} |T(s_k)|} \tag{3}$$

In order to both contextualize and personalize the selection of a service, apart from the tag cloud of a service in the system (the one which characterize the context of the service globally), we also need to know the contextual tag cloud of a service form the point of view of a specific user, that is, the contexts where the user invokes the service. For that, we define the personal $TC_S(s,u) = \{t, w(s,t)\}$ such that $(u,t,s) \in Y$ and the weight is computed as described above. Finally the tag cloud of the actual context is denoted by $TC(home)$ and obtained from the data and metadata provided for all the devices at home, the tags in the tag cloud $TC(home)$ are weighted accordingly.

5.2 Creating a weighted graph for contextualization

In order to properly compare the tag clouds of the actual context at home $TC(home)$ and the one which records the context where a service has been previously accessed ($TC_S(s)$ or $TC_S(s,u)$), we define a weighted graph TG_C for the folksonomy F_C (respectively TG_C^u for the personomy P_C^u). This weighted graph is created from the service contextual tag clouds $TC_s(s)$ in the system (respectively $TC_S(s,u)$ for a specific user u).

TG_C is an undirected graph where the nodes are the tags ever used in the system and the arcs are the relationships between the tags they link. The relationships are calculated from the number of times the contextual tags appear together in a contextual tag cloud, the weights of the tags in this tag cloud, as well as the index of popularity (IOP) of those services described by both contextual tags at the same time. We define the relationship between two tags (t_i and t_j) with respect to one single service s (denoted as $r_{ij}(s)$), as the geometric means between the weights of both tags in $T(s)$.

$$r_{ij}(s) = \sqrt{w(s,t_i)\, w(s,t_j)} \tag{4}$$

To obtain the total relationship between t_i and t_j (r_{ij}), we should take into account their partial relationships with respect to each service in the smart home and appropriately weight them according to its IOP.

$$r_{ij} = \frac{\sum_{\forall s / t_i \vee t_j \in T(s)} IOP(s)\ r_{ij}(s)}{\sum_{\forall s / t_i \vee t_j \in T(s)} IOP(s)} \tag{5}$$

5.3 Tag cloud comparison

The selection process need an algorithm of tag cloud comparison. Concretely, the contextual tag cloud which describes the actual context $TC(home)$ has to be compared with the information in the contextual folksonomy in order to decide which service of a specific category is more suitable. As it was mentioned before, in this decision we can vary the quantity of collective intelligence taken into account, that is, using only the information in the user personomy or using the information in the system folksonomy. In this section we propose a metric for comparing contextual tag clouds.

The simplest way to measure this value would be to count the number of coincident tags in both contextual tag clouds, i.e., the higher the number of coincident tags, the higher the degree of relationship between the actual context at home and the contexts where a specific service has been used. Besides, to take into account the relative importance of the coincident tag in the tag cloud, instead of adding 1 for each coincident tag, it is better to add the means of their weights in both tag clouds. The relationships is then calculated like follows:

$$R_0(TC_S(s), TC(home)) = \sum_{\forall i / t_i \in |T(s) \cap T(home)|} \sqrt{w(s, t_i)\ w(home, t_i)} \tag{6}$$

But this approach does not take into account the relationships between the tags in the weighted graph of the folksonomy TG_C. Although a tag that belongs to the first tag cloud do not appear in the second tag cloud (or appears with a lower weight), it can be closely related to the rest of the tags of the second tag cloud. To take this fact into account, we do not only consider the number of coincident tags and their weights (*direct relationship*, $R_0(TC_S(s), TC(home))$, but also the degree of relationship between the tags of both tag clouds (*one-hop relationship*, $R_1(TC_S(s), TC(home))$). In this manner, the total relationship is calculated as follows:

$$R(T_S(s), TC(home)) = \alpha\ R_0(T_S(s), TC(home)) + (1 - \alpha)\ R_1(T_S(s), TC(home)) \tag{7}$$

where the parameter $\alpha \in [0, 1]$ is used to assign more or less importance to the direct or one-hop relationships. The one-hop relationship is calculated by adding the degree of relationship of each pair of possible tags (each tag of the pair belongs to one of the tag clouds) adjusted by the means of their weight.

$$R_1(T_S(s), TC(home)) = \sum_{\substack{\forall i / t_i \in TC_S(s) \\ \forall j / t_j \in TC_(home)}} \sqrt{w(s, t_i)\ w(home, t_j)}\quad r_{ij} \tag{8}$$

Finally, if we want to compare the context at home $TC(home)$ with the context tag cloud of a service from the point of view of a specific user $TC_S(s,u)$, that is in the personomy, the relationships are calculated in a similar way but taking into account the relationships in the weighted graph TG_C^u.

6. Two forms of personalization: the service and its invocation

Although the above preference model allows personalization strategies for devices, providers, etc., we focus on the two most common applications: personalizing the selection of the service and its invocation personalization. Our proposal to provide a context-aware personalization for OSGi services is based on the OWL-OS framework. It defines two logic layers to automatically select (according to the user's preferences and the contextual information) which is the most adequate service and how must it be invoked, respectively. The first stage decides the service in a specific category (*OSGiServiceProfile* entity in OWL-OS); for instance, the most suitable service for playing music. The second one decides how to invoke (parametrisation) the specific service (*OSGiServiceInvocation*); for instance, the most suitable volume when playing music (see Fig. 6).

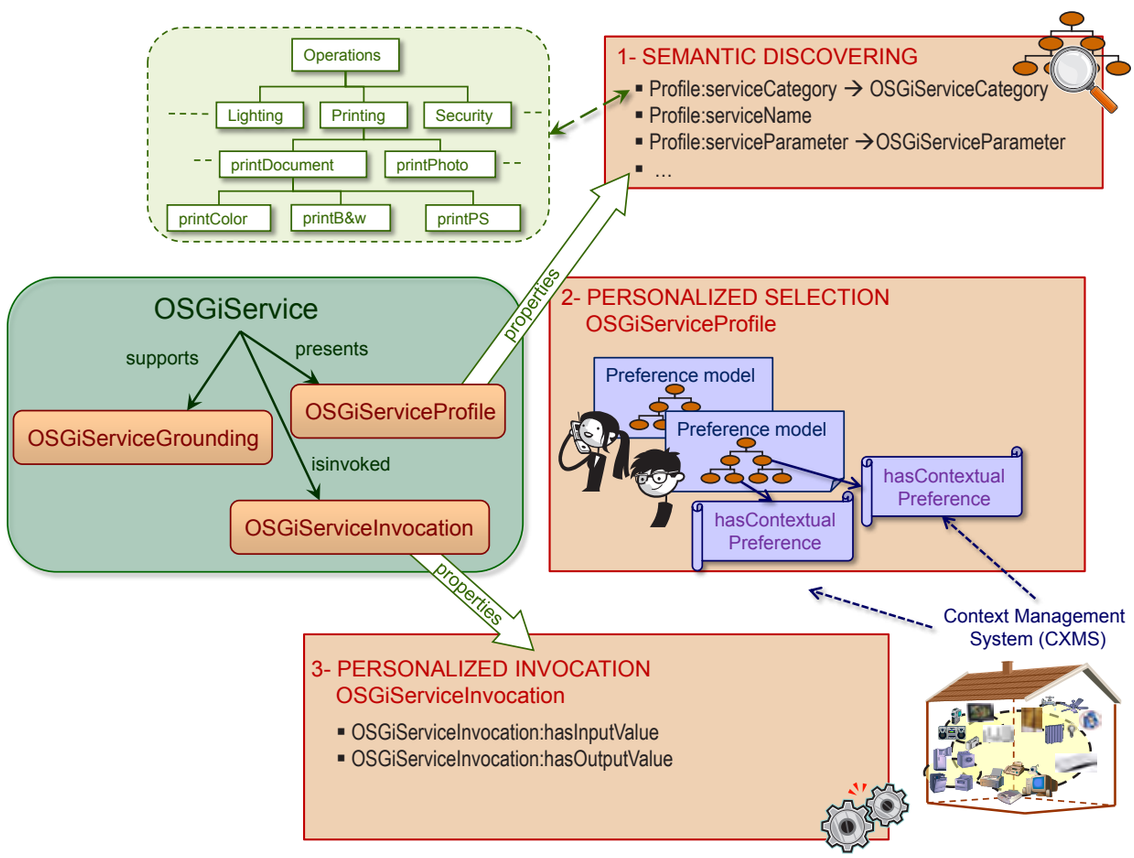


Fig. 6. Using Semantic OSGi for personalizing OSGi services in a context-aware solution

6.1 Context-aware and personalized selection

Service selection can be personalized by using the information in the preference model to establishes the most appealing service for the user U among those which meet the discovering criteria (*OSGiServiceCategory* and *OSGiServiceParameter*) in the current context C. With this aim, we have added another method inspired in the OSGi original ones:

```
semGetServiceReferences(String OntURI, String Cat, String semFilter)
```

This method includes a filtering criterion (`semanticFilter`) based on the SWRL Query instead of LDAP. Thus, the *semFilter* is an OWL query about individuals pertaining to the category specified in the method's second parameter.

From the client's perspective, the same method (`semGetServiceReference` or `semGetServiceReferences`) is used. From the Framework's perspective, the service selection process is quite different and proceeds according to the following steps:

[1] Semantic matching for the service s: First, the semantic OSGi Registry looks for those services that match the category (`Cat`) and satisfy the (`semFilter`), if applicable.

[2] Context-aware matching for the user u in the context $TC(home)$: Now, it starts the personalization part of the service selection using the preference model of the active user u and taking into account the context $TC(home)$. For each service s from step 1, the CXMS receives the `hasContextualPreference` properties in the ontology to do the following steps:

[2.1] Contextual Preference for the user: An index of contextual preference for the user u ($\mathcal{I}_{CP}(s,u)$) is calculated which reflects the preference of the user u about the service s but adjusted by the similarity between the actual context at home and the previous contexts in which the service s has been invoked by the user u:

$$\mathcal{I}_{CP}(s,u) = DOP(s,u)\ R(T_S(s,u), TC(home)) \tag{9}$$

The above parameters are registered in the properties `hasDOP` and `hasContext`) of the service profile instance s for the user u.

[2.2] Contextual Preference for the community: An index of contextual preference for the community ($\mathcal{I}_{CP}(s)$) is calculated which reflects the preference of the community about the service s but adjusted by the similarity between the actual context at home and the previous contexts in which the service s has been invoked by the community:

$$\mathcal{I}_{CP}(s) = DOP(s,u)\ R(T_S(s), TC(home)) \tag{10}$$

The above parameters are registered in the properties `hasDOP` and `hasContext`) of the service profile instance s for the user community.

[3] Context-aware an personalized selection: The semantic OSGi Registry selects the most adequate service, from the ones filtered in step 1, choosing the service s having the highest contextual preference. For this a global index of contextual preference $\mathbf{I}_{CP}$ is defined. This new index allows to select the quantity of collective intelligence to be incorporated in the selection process by adjusting the β parameter in the following equation:

$$\mathbf{I}_{CP}(s) = \beta\, \mathcal{I}_{CP}(s,u) \;+\; (1-\beta)\, \mathcal{I}_{CP}(s) \tag{11}$$

6.2 Context-aware and personalized invocation

One step beyond personalizing service selection is adapting the invocation of that service (typically the values for input parameters) according to the user, his/her habits and the context. Thus, it is possible to capture behaviors like the following: when playing music (the service) John (the user) selects, during the day (context), 90% of the times (*DOP*) high volume; and, at night (context), 80% of the times (*DOP*) low volume. With this aim, and as there is no notion of service execution on OWL-S nor OWL-OS, we introduce the class `OSGiServiceInvocation` in OWL-OS. As Fig. 6 shows, the main properties in this class are: (i) `hasInputValue` and `hasOutputValue`, instances of `Binding` OWL-S classes[3]; and (ii) `hasContextualPreference`.

In order to allow the implementation of the behavior above, we have added specific methods to the OSGi framework: `getInputValue` and `getOutputValue`. The former returns, for a specific service, the preferred value (greater index of contextual preference $\mathbf{I}_{CP}$) among those *OSGiServiceInvocation* entities in the preference model of the user u or of the community depending on the similarity with the current context of the smart home. The latter has the same behavior but referred to an output value. Despite the fact that the application of the preferred output value remains open, it can support, for instance, an automated alarm system which notifies in case of output values different from the preferred ones.

7. Implementation

The prototype of this proposal has been developed over OSCAR (Open Source Container ARchitecture, `http://oscar.objectweb.org`), an open software implementation of the OSGi framework. We have added a semantic version of the OSCAR registry which interprets the new bundle manifests and manages the OWL-OS ontology to search (querying the ontology) and to register services (populating the ontology). Besides, the semantic version of the OSCAR registry maintains the preference models for the house's inhabitants, so it also modifies the services' properties for each profile, like the DOP or the context. To manage the local OWL-OS ontology, the profiles and provide the semantic OSGi services, we use the Protégé OWL API (`http://protege.stanford.edu/plugins/owl/api/`). This open-source Java library provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning. The CXMS we have designed has been integrated as an OSGi service in the OWL-OS platform. This CXMS manages the context folksonomy and the metrics presented in this paper for modelling and comparing context information.

[3] A binding object has two properties the name of the variable, `toVar`, and the specification of a value for which we suggest `ValueData`.

8. Related Work

Building smart homes entails integrating different computing technologies, such as ubiquitous computing, context-aware computing or home automation technology. This combination supports automatic interactions among residents, computer-equipped devices and the home environment.

Context Modeling entails to represent "any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves" (Dey, 2001). There are different approaches to organize context knowledge (Strang & Linnhoff-Popien, 2004): from the simplest technique based on a list of attributes in a *key-value* manner to the more sophisticated *ontology-based models*, where ontological structures provide an uniform way for specifying the model's core concepts as well as enabling contextual knowledge sharing and reuse in an ubiquitous computing system; without forgetting graphical, object-oriented or logic-based models. However, having a successful context modeling usually requires of combined strategies. Specially relevant for the smart home, the authors' approach in (Oh et al., 2006) consists of defining any context at home by five concepts (4W1H) : who (resident identification), what (identifies the event that should happens), where (resident's location), when (occurring time) and how (the way the event should develop within these conditions).

OSGi and the smart home. Previous approaches have tried to promote smart spaces by using OSGi, like in (Lee et al., 2003), whose authors propose using OSGi as a suitable infrastructure to integrate various devices and sensors to provide pervasive computing environments. However, they do not resolve the problem of service search and invocation. In (Dobrev et al., 2002) these two problems are directly tackled but not from a semantic perspective. On another hand, the authors of (Gu et al., 2004) propose to define a semantic environment for describing contextual information to be used by context-aware applications. However, OSGi is only used as a support layer, without improving the OSGi framework at all. After an exhaustive revision of the state-of-the-art, we have not found any proposal about integrating semantic reasoning nor personalization within the OSGi framework.

Ontological descriptions for the smart home. Some initiatives have came up in this area. UbisWorld (Heckmann, 2003) is used to represent parts of the real world (a house, a shop, etc.) and all the individuals and elements belonging to it. Its knowledge is modeled by two ontologies which are under development: GUMO (General User Model Ontology) and UbisOntology, the ontology for ubiquitous computing. SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) (Chen et al., 2004) is another relevant approach. It consists of two set of ontologies: SOUPA *Core* to define generic vocabularies; and SOUPA *Extension* to define additional vocabularies for specific applications. However, no focus has been on the semantic description of the in-home services. For this, we propose to take advantage of the previous efforts made in the Semantic Web field, where the appropriate semantic description of services is the main point. In this area, the most salient initiatives to describe Semantic Web Services are WSMO (Web Service Modeling Ontology) (*D2v1.2 Web Service Modeling Ontology*, 2005) and OWL-S (*OWL-S: Semantic Markup for Web Services. 1.1 Release*, 2004). Although both

approaches cover the same field with the same goals trusting ontologies to get them, OWL-S[4] is considered to be clearly more mature in certain aspects and expressive enough to be applied in the smart home environment.

Collaborative tagging and folksonomies Collaborative or social tagging allows users to tag contents and share them in such a way that they can categorize not only content that they themselves have added, but also content added by other users Golder & Huberman (2006). Sites commonly cited as examples of collaborative tagging are Del.icio.us , for the tagging of Web pages, or Flickr, for the tagging of photos. In order to define the structures generated by this type of classification, in 2004 Thomas Vander Wal coined the term folksonomy citepdef-folksonomy. At the same time he established two different types of folksonomies: broad folksonomies, constructed as a result of tagging contents in systems in which any user may tag all the content in the system (as in Del.icio.us), while narrow folksonomies are the result of tagging systems in which only a small number of users may tag content (for example, Flickr, where only the author of the photos and users designated as friends of the author may assign tags to them).

Folksonomies, whether wide or narrow, are structures that can be represented as an undirected graph in which the nodes are the various tags assigned in the system and the transitions are the relationships between two nodes that are joined. These relationships are assigned a weighting which will depend on the number of times tags describing a content item appear together: the higher the frequency, the higher the weighting Michlmayr et al. (2007). A structure of this type lends itself to be used to establish relationships between elements of the system. Some works in the literature have explored a folksonomy-based solution for personalization or recommendation in several domains. The proposal in Niwa et al. (2006) makes use of users' favourite Web pages (obtained from a collaborative tagging system for Web bookmarks such as Del.icio.us) and their tags to recommend new Web pages. To do this it calculates the affinity between users and Web pages, grouping related tags together and finding out which Web pages are the most appropriate to each group. Next it calculates the affinity of the user with the tags of the groups to determine which Web pages are the most appropriate to him or her. Folksonomies can also be used to calculate the relevance of content shown in information retrieval systems; for example, the approach set out in Hotho, Jäschke, Schmitz & Stumme (2006a)Hotho, Jäschke, Schmitz & Stumme (2006) describes an algorithm known as FolkRank (based on the idea of Google's Page-Rank), whereby a resource tagged as important by important users is considered to be important. The principle is applied to both users and tags. Specia & Motta (2007) presents another work in which tags are grouped together according to the relationship between one another. In this case, in order to establish similarity between tags, the system takes into account not only the number of times that two tags appear together but also the number of times that the first of the two tags is used together with tags that, in turn, are used together with the second of the two. Finally, these groups are refined by searching for each pair of tags on a number of different semantic search engines to confirm the relationship between them.

[4] The OWL-S coalition is currently working in an upcoming release 1.2 of OWL-S, whose details are available at `http://www.ai.sri.com/daml/services/owl-s/1.2`.

9. Conclusions and future work

The primary reference architecture in the OSGi specification is based on a model where a operator manages a potentially large network of service platforms. It assumes that the service platforms are fully controlled by the operator and used to run services from many different services providers. In this scenario, we have shown that the actual service discovery mechanisms in OSGi is insufficient. In the pursuit of a really open and interoperable residential gateway, we propose the semantic description and discovery of the services in the OSGi domain. At this respect, we have defined OWL-OS, a sub-ontology of OWL-S which allows making a simple semantic search of services based on a categorized structure. Despite we propose *operations-at-home* as the primary structure to classify the OSGi services, OWL-OS allows an OSGi service to be semantically described according to different ontological structures. These ontological structures would be downloaded on demand from the service provider. Finally, note the Semantic OSGi Framework enhance the OSGi standard, without breaking it; i.e. any non-semantic bundle can work properly within this framework, although it is not able to take advantage of the semantic reasoning for service obtaining. Moreover a clear benefit of the new Semantic OSGi platform is the possibility of supporting the automation of OSGi services composition (Díaz Redondo et al., 2007).

In the field of service selection, it would be possible to supplement the Semantic Registry with different specialized software agents which takes into account other factors (ambient intelligence) to automate the service selection. In this paper we have investigated the potential of combining a personalization agent and a context management system. On its own, the personalization agent contribute to decisions on service selection by using a preference model storing historic information about preferred devices and services. On the other hand, the context management system allows the OWL-OS Framework to be aware of the particular context (time, location, mood, etc) and to react by discovering services which meet that context. Beyond their independent behavior, connecting the personalization and context-aware agent across an ontological base, OWL-OS in our case, allow the OWL-OS framework to support scenarios as the one included in this paper. In general, scenarios where the selection of the service depends not only on who uses the service but when, where or even why uses the service. Although this paper only discusses two forms of context-aware personalization in the smart home (service selection and parametrisation), some other forms are possible with the general preference model we propose. For instance, provider selection can be personalized by capturing the preferences of the user.

Apart from the general approach, one of the main contributions of this work is the effort to provide a flexible context model which fit in well with an extremely heterogeneous environment. With this premise, we provide a collaborative approach to context, in fact we can say that the solution is collaborative in two senses. Firstly, it is a collaborative model of context since all the devices, services, sensors, inhabitants, etc. at home form a pluriarchy without any controlled vocabulary. Secondly, it is also a collaborative model for preferences since the service selection takes into account not only the preferences of the user but even the preferences of a smart home community, possible a community established by the service provider. At this respect, in this paper we have explored algorithms of service selection which rely on tag cloud

comparison of contexts, but it is also possible to inspect the similarity between users by comparing user's preference model. Considering the similarity among users, according to their behavior at home, allows the selection process to incorporate more sophisticated strategies of collaborative filtering.

Also as part of our future research, we are working on supporting multi-user preference model. The human presence simulator in this paper is already a multi-user system whose aim is simulating the behaviors of all the inhabitants living in the smart home. The solution to this problem is far from being trivial, because an approach which simply merge the habits of all the inhabitants obviates the relationships and even dependencies among users at home. An adequate multi-user personalization system has to distinguish among the user preferences and cope with conflict resolution.

10. References

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The Semantic Web, *Scientific American* pp. 35–43.

Chen, H., Perich, F., Finin, T. & Joshi, A. (2004). SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications, *Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services*.

D2v1.2 Web Service Modeling Ontology (2005). The WSMO group.
URL: *http://www.wsmo.org/TR/d2/v1.2/*

Dey, A. K. (2001). Understanding and using context, *Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing* **5**(1).

Díaz Redondo, R. P., Fernández Vilas, A., Cabrer, M. R., Pazos Arias, J. J. & Rey López, M. (2007). Enhancing Residential Gateways: OSGi Services Composition, *IEEE Transactions on Consumer Electronics* **53**(1): 87–95.

Díaz Redondo, R. P., Fernández Vilas, A., Ramos Cabrer, M., Pazos Arias, J. J., García Duque, J. & Gil Solla, A. (2008). Enhancing Residential Gateways: a Semantic OSGi Platform, *IEEE Intelligent Systems Journal* **23**(1): 32–40.

Dobrev, P., Famolari, D., Kurzke, C. & Miller, B. A. (2002). Device and Service Discovery in Home Networks with OSGi, *IEEE Communications Magazine* **40**(8): 86–92.

Golder, S. A. & Huberman, B. A. (2006). Usage patterns of collaborative tagging systems, *J. Information Science* **32**(2): 198–208.

Gu, T., Pung, H. K. & Zhang, D. Q. (2004). Toward an OSGi-based infrastructure for context-aware applications, *IEEE Pervasive Computing* **3**(4): 66–74.

Heckmann, D. (2003). Integrating Privacy Aspects into Ubiquitous Computing: A Basic User Interface for Personalization, *Artificial Intelligence in Mobile System Workshop at Ubicomp (AIMS)*.

Hotho, A., Jäschke, R., Schmitz, C. & Stumme, G. (2006a). Folkrank: A ranking algorithm for folksonomies, *Proc. FGIR 2006*.

Hotho, A., Jaschke, R., Schmitz, C. & Stumme, G. (2006b). Information retrieval in folksonomies: Search and ranking, *The Semantic Web: Research and Applications*, Springer, pp. 411–426.

Hotho, A., Jäschke, R., Schmitz, C. & Stumme, G. (2006). Information retrieval in folksonomies: Search and ranking, *Proceedings of the 3rd European Semantic Web Conference*, LNCS, Springer, Budva, Montenegro, pp. 411–426.

Howes, T. (1996). A String Representation of LDAP Search Filters, *Technical Report RFC 1960*, University of Michigan.

Lee, C., Nordstedt, D. & Helal, S. (2003). Enabling Smart Spaces with OSGi, *IEEE Pervasive Computing* **2**(3): 89–94.

Meyer, S. & Rakotonirainy, A. (2003). A survey of research on context-aware homes, *CRPITS '03: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003*, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 159–168.

Michlmayr, E., Cayzer, S. & Shabajee, P. (2007). Add-A-Tag: Learning Adaptive User Profiles from Bookmark Collections, *Proceedings of the 1st International Conference on Weblogs and Social Media (ICWSM'06)*.

Mizzaro, S., Nazzi, E. & Vassena, L. (2009). Collaborative annotation for context-aware retrieval, *ESAIR '09: Proceedings of the WSDM '09 Workshop on Exploiting Semantic Annotations in Information Retrieval*, ACM, pp. 42–45.

Niwa, S., Doi, T. & Honiden, S. (2006). Web page recommender system based on folksonomy mining for itng ’06 submissions, pp. 388–393.

Oh, Y. S., Yoon, H. S. & Woo, W. T. (2006). Simulating Contest-Aware Systems based on Personal Devices, *Proceedings of the International Symposium on Ubiquitous VR*, pp. 49–52.

OSGi Service Platform, Core Specification, Release 4 (2005). The OSGi Alliance.
URL: *http://www.osgi.org*

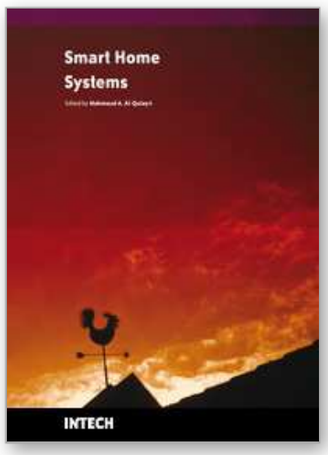
OWL-S: Semantic Markup for Web Services. 1.1 Release (2004). The OWL Services Coalition.
URL: *http://www.daml.org/services/owl-s/1.1/*

Specia, L. & Motta, E. (2007). Integrating folksonomies with the semantic web, *ESWC '07: Proceedings of the 4th European conference on The Semantic Web*, Springer-Verlag, Berlin, Heidelberg, pp. 624–639.

Strang, T. & Linnhoff-Popien, C. (2004). A context modeling survey, *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing*.

Wal, T. V. (2007). Folksonomy coinage and definition.
URL: *http://vanderwal.net/folksonomy.html*

Zimmermann, A., Specht, M. & Lorenz, A. (2005). Personalization and context management, *User Modeling and User-Adapted Interaction* **15**(3-4): 275–302.



Smart Home Systems

Edited by Mahmoud A. Al-Qutayri

ISBN 978-953-307-050-6
Hard cover, 194 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

Smart homes are intelligent environments that interact dynamically and respond readily in an adaptive manner to the needs of the occupants and changes in the ambient conditions. The realization of systems that support the smart homes concept requires integration of technologies from different fields. Among the challenges that the designers face is to make all the components of the system interact in a seamless, reliable and secure manner. Another major challenge is to design the smart home in a way that takes into account the way humans live and interact. This later aspect requires input from the humanities and social sciences fields. The need for input from diverse fields of knowledge reflects the multidisciplinary nature of the research and development effort required to realize smart homes that are acceptable to the general public. The applications that can be supported by a smart home are very wide and their degree of sophistication depends on the underlying technology used. Some of the application areas include monitoring and control of appliances, security, telemedicine, entertainment, location based services, care for children and the elderly… etc. This book consists of eleven chapters that cover various aspects of smart home systems.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen