Semantically Enabled Framework for User Centric Profile Description, Search and Match

João Moura[1,⋆], Paulo Chainho[1,⋆], and C.V. Damásio[2,⋆⋆]

[1] Portugal Telecom Inovação, Tagus Park Edifício Tecnologia II,
31 2780 - 920 Porto Salvo - Portugal
[2] Centro Inteligência Artificial, Universidade Nova de Lisboa, Portugal
cd@di.fct.unl.pt

Abstract. With the advent of User Centric network paradigms like IP Multimedia Subsystem (IMS) and Web 2.0 the user must be able to describe both him self and the contents he produces in an unambiguous way as Service Oriented Architectures (SOA) promotes the emergence of more and more services. The user will be challenged to survive in a wild and wide service ecosystem. In this paper we present the Matcher network functionality that aims to provide guidance to the user in such a rich but complex service ecosystem. The Matcher is a framework not only to allow the user and service providers to produce semantically enabled (user, service and content) profiles and match these profiles, but also to obtain useful results using simple tags and semantically enabled tags - statements.

Keywords: Semantic Web, Service Discovery, IMS, SOA, WEB 2.0, XDM.

1 Introduction

Today's WWW has well over 10 billion pages [1], but the vast majority of them are in human-readable format only (e.g., HTML). The massive proliferation of Broadband accesses exponentially raises the number of reachable users and its devices. On the other hand the emerging Telecom SOA (TSOA) also aims to achieve a significant amount of services notably user generated services [13]. In such a complex environment the ability to consume the most appropriate service, to find the right person or to get the right content will become a key differentiator among service providers. In the meanwhile, researchers have created the vision of the Semantic Web [3], where data has structure and ontologies describe the semantics of the data. When data is marked up using ontologies, services can better understand the semantics and therefore more intelligently locate and integrate data for a wide variety of tasks, namely community and service recommendations to matching users. The Semantic Web thus offers a compelling vision, but it also raises many difficult challenges mainly because of the lack of a coherent and standardized universal model. Ontology mapping techniques like GLUE [5] can be of some

⋆ Partially funded by the European Union under contract IST-034101. Integrated in the Opuce project. OPUCE is an Integrated Project of the 6th Framework Programme, Priority IST.

⋆⋆ Partially supported by European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net).

E. Di Nitto and M. Ripeanu (Eds.): ICSOC 2007 Workshops, LNCS 4907, pp. 326–335, 2009.

help. In this paper we propose a Matcher network module, allowing the regular user to describe him self and to have the power to find any service or other component that he is implicitly interested at. The key concept of our framework is the profiling feature. Everything can be described with an XML profile, what we must do is to provide the match functionality. To provide this functionality, the Matcher is composed by different pluggable Match Providers each one using a different Matching algorithm. This paper is mainly focused on the Semantic Match Provider that deals with semantic ambiguity by combining:

- Semantic matching via statement description based on RDF [10] and OWL [16]; Both General and Domain specific taxonomies available for tag classification allowing *Semantic inferences* performed with a reasoner;
- *First-Order Logic* (FOL) based querying composition and result aggregation; Result mining and validation;

Specifically, we use transitive closure algorithms to provide a statement editor; reasoners to provide semantic inferences and to compute an overall semantic similarity score between concepts. In addition we integrate semantic matching with an indexing method, which we call tag hashing, enabling fast lookup of semantically related concepts. Finally, the Matcher deployment in emerging SOA and IMS-like network environments is facilitated by the extensive usage of standardized Service Enablers mainly the Open Mobile Alliance's (OMA) [2] XML Document Management (XDM) [11] Server. This paper focuses on describing the architecture and the overall functionalities rather than going in depth over the theoretic and implementation issues. It is divided in two main sections:

- **Technical Overview.** The matcher architecture and related algorithms, including a semantic match provider and the match broker, are described;
- **Matcher in Action.** The integration of the platform prototype in real telecommunication architectures is discussed and some examples are presented.

2 Technical Overview

In response to the lack of widespread usage of semantic technologies on the web, our framework proposes the ability to easily tag profiles with semantic description, allowing more efficient and time saving matches when compared to unstructured and index free architectures. Our matcher is divided in three main sub modules, the Match Broker, the Match Providers (e.g. semantic match provider) and the Profile Deployment Manager.

In order to describe services and resources, wikipedia entries are used as common vocabulary among users and service providers. This provides an easy and user-friendly way for describing concepts and resources, with multi-lingual support as well. Currently, the ontologies are expressed in the Ontology Web Language [16], and matching queries are specified using only class description involving class identifiers, intersections, union and complements. In theoretical terms, this problem can be viewed as instance checking of individuals with respect to concepts. Particularly, we are interested is the data complexity of testing whether $KB \models C(a)$ where KB is a knowledge base

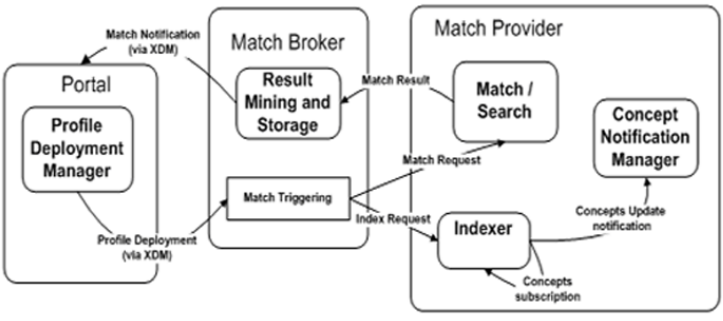

Fig. 1. Functional Architecture

in OWL, C is a concept name, and a an individual (for instance see [6]). It is shown that without transitive roles, this problem is coNP-complete [12], in particular for the very simple ALC language. However, by limiting the constructs used in the Knowledge Bases (ontologies) reasoning can be reduced to polynomial time [4]. This justifies the limitations used.

2.1 Profile Deployment Manager (PDM)

The PDM provides Semantic Classification and Query Composition functionalities allowing to build statements by presenting a a very simple statement editing tool to the user. It fetches the available ontologies (domain specific) from the XDM server and presents a tag classification tool to the user (e.g. SafariRide @ Safari)

These statements can be composed using the basic logic operators *and*, *or* and *not* and form queries that are performed by deploying a set of statements. A profile must be deployed on the XDM, using a XML Configuration Access Protocol (XCAP) interface [7], which will send a notification to the Matcher module, originating the deployment process. This profile consists of standard information like name, id, address etc and also the description which is a set of owl statements, simple tags and keywords and codified in XML.

```
<profile xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <profileID>LX</profileID>
 <profileType>0</profileType> (.......)
 <interests>  </keywords> <tags> <tag>Portugal</tag> </tags>
    <statements> <statement>
      <ontology>travel.owl</ontology>
       <body>
        <Capital rdf:about="http://wikipedia.org/wiki/Lisbon"/>
       </body>
     </statement> </statements>
 </interests>
</profile>
```

Most sections of the XML based profile are omitted because not all of them are *directly* used for matching purposes, but rather for further description.

2.2 Match Broker

The Match Broker is the core of the framework. It is responsible to, as much as possible, abstract the providers from the details of storage and networking allowing for changes to be made locally. It is composed by two main functionalities:

Match Triggering. This functionality handles the events that trigger the Match process, forwarding them to the appropriate Match Providers. According to our specification, the Matcher is triggered by the XDM server's Session Initiation Protocol (SIP) notifier [8]. It assumes that the subscriptions to profile deployments were already done by the Matcher. Moreover, any change on the profile description will also trigger an event fire to the Match module, containing the changes, that will be handled virtually the same way as a new profile deployment is. Those changes include both added and deleted tags. For each added tag, the engine must add the profile id to the list identified by the tag name appended with the profile type. For each removed tag, the corresponding list is updated. This process allows for easy, fast and really localized updates on the information needed to perform the matches - *tag hashing*.

Result Mining and Storage. This functionality collects, aggregates and stores the Match results. Match results are identified by the profile ID who requested the match; aggregated by matching concept and weighted by an *actually inserted concept* to *inferred matching concept* distance. In order to present the results in an ordered fashion, a *leaf to concept classification class* metric and *nearest match* metric can then be applied:

Leaf to Concept Classification Class (LCCC) metric – Concepts can be classified as being instances of a given ontology class, the distance between that class and the nearest leaf on the ontology's subclass tree is the LCCC distance.

Nearest Match metric – Concepts can be classified as belonging to a given Class on Ontology. The more concepts two profiles have in common the more they can be considered as matching. So, profiles with the biggest number of matching concepts should be listed first in the match results.

Afterwards, several approaches can be taken in order to provide the result to whoever asked for it by executing a given composed query, finding users with the most resemblance or ordering the results by one of the presented metrics:

Mixed metric mining – In some cases more isn't better. If a profile matches some other profiles but with different *Leaf to Concept Classification Class* result distances to each of them, then the matches should ordered.

$$profileA\,\{a2, b1, \ldots, p4, q0, r0\} \quad profileB\,\{p4, q0, r0\} \quad profileC\,\{a2, p4, q0\}$$

On this example $profileA(pA)$ matches $profileB(pB)$ and $profileC(pB)$ on three concepts: p, q, r and a, p, and q respectively who are weighted according to their distance to a leaf. Hence, q and r are described as a belonging to a leaf Class. So the matches are tied. The way they can be ordered is by using the *LCCC* metric where the best match would be calculated by adding up each concept's classification for each

match. For pA match with pB the concepts involved are p, q and r. Since they are mapped on the same ontology the weight is the same on each profile so the match accuracy would be: three weighted with 4+0+0. Between pA and pC the accuracy is three weighted with 2+4+0.

Query Evaluation Mining – From the queries that are composed on the Deployment manager, the actual search or match is performed individually for each statement, then the results are composed by creating and evaluating an abstract syntax tree(AST) [15] with operators and match results. The final result of the query, is the result of evaluating the tree, and is stored in the XDMS via XCAP.

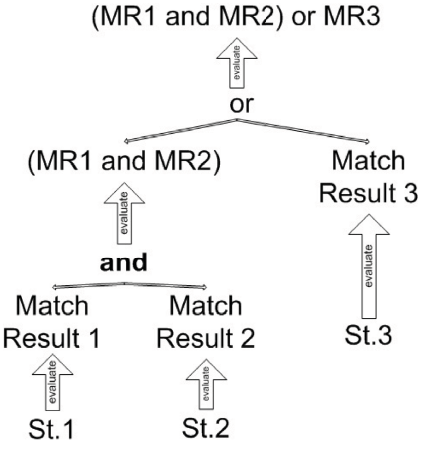

Fig. 2. Query abstract syntax tree

The Statements St1 to St3 are all evaluated separately and the partial Match results MR1 to MR3 are obtained. They are aggregated by evaluating the FOL operators (bottom up) until the final result is obtained. For simplification the parenthesis expansion is omitted without loss of generalization.

2.3 Semantic Match Provider

The Semantic Match Provider is composed by two main functionalities providing the actual matching process: the Indexer requires profiles to be deployed marked with indexing request; the Match / Search functionality allows querying, also by deploying profiles marked with a match request.

Indexer Functionality. At the Matcher's heart, tag hashing consists of having a list of all profiles that match any given concepts. Thus, several lists are stored, creating an index with linear access time complexity for each concept. When deploying a profile containing semantic enabled tags or simple tags, the profile id is indexed, using one for each concept and concept inferred using jena owl micro reasoner [9].

Example 1. Concept1 @ A states that the list refers to Concept1 URL classified as belonging to class A on a given ontology, identified by the tag name appended with the type of profile being deployed, e.g. ID6-T2.

These lists contain all the profile ids of already deployed profiles from that type that were described using that concept. Other similarly built lists may also exist for the same concept only that appended with a different profile type. This indexing method is actually the basis of the match/query engine.

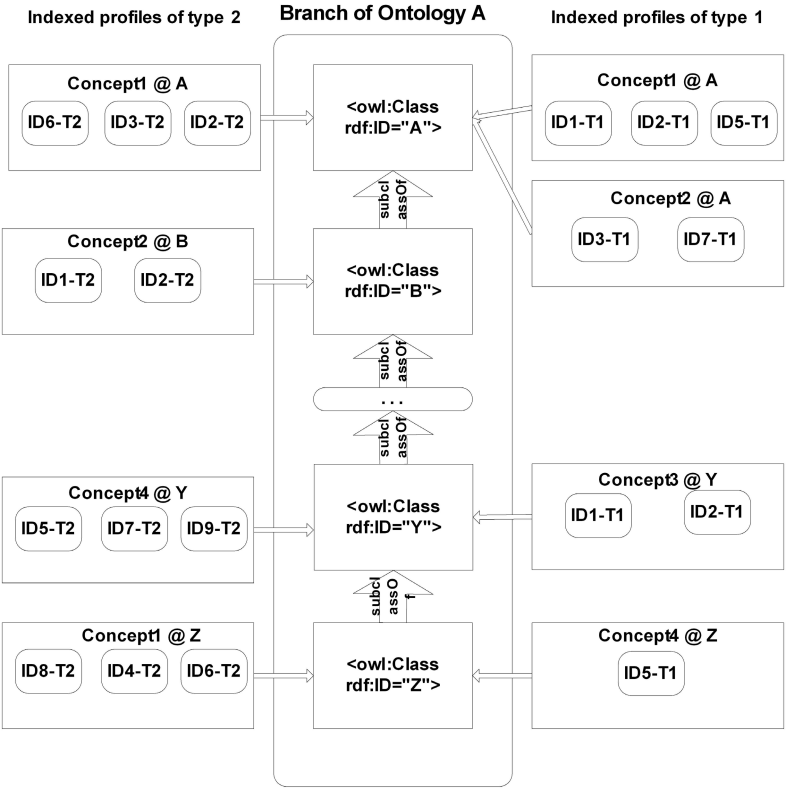

Fig. 3. Tag Hashing

During this indexing procedure, the profile owner implicitly subscribes to the lists of concepts he used to describe the profile he deployed. There is one file on the XDM server for each of the concepts being handled. It contains all the profile ids, of each type, that match the concept. So, for each concept, a notification subscription to any changes on that concept index file is done (see the CNM functionality from the Broker).

Match/ Search Functionality. When the actual match is requested, via the deployment of a profile containing a match flag, the index lists matching each concept are

aggregated. If a concept is directly matched then it is weighted with 0. Otherwise, if it is matched via an inference done with the jena reasoner, then it is marked with the distance on the subclass tree of the used ontology, between the concept actually contained by the profile and the inferred class of the matching inferred concept.

Example 2. The profile marked with the concept `<Safari rdf:ID="SafariRide"/>` in Figure 4 should match the inferred `<Adventure rdf:ID="SafariRide">`, `<Sightseeing rdf:ID="SafariRide">` and `<Activity rdf:ID="SafariRide">`, even if weighted with different values according to an appropriate metric.

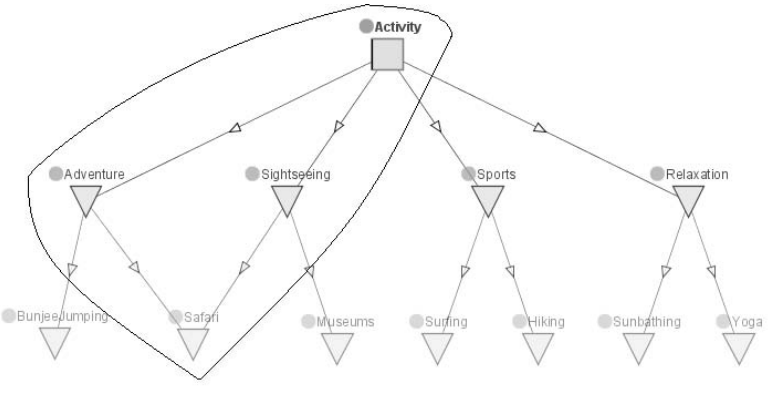

Fig. 4. Travel.owl branch

This figure represents a branch from a travel taxonomy. Any Safari should be considered an activity of adventure and sightseeing.

Concept Notification Manager. When the profile id from a newly deployed profile is added to each of the concept lists (as described), the Semantic Match Provider is notified according to previous subscription performed by the Indexer. This notification manager makes sure that the user doesn't get any double notifications coming from all the deployments associated at a concept that are triggered by the inference procedure. E.g. in Figure 4, a deployment of a concept classified as a Safari triggers the deployment of the same concept marked as all of the super classes. In fact the user is only interested in receiving a notification of concepts marked with the smallest *Nearest-Match* metric.

3 The Matcher in Action

Given the proposed web service interface, the presented solution can have it's place in several architectures almost effortlessly. A Matcher prototype derived from our specification was implemented based on J2EE and Jain SLEE technologies [14].

JSLEE is an asynchronous, very performance and low latency prone container, widely used in the telecom industry.

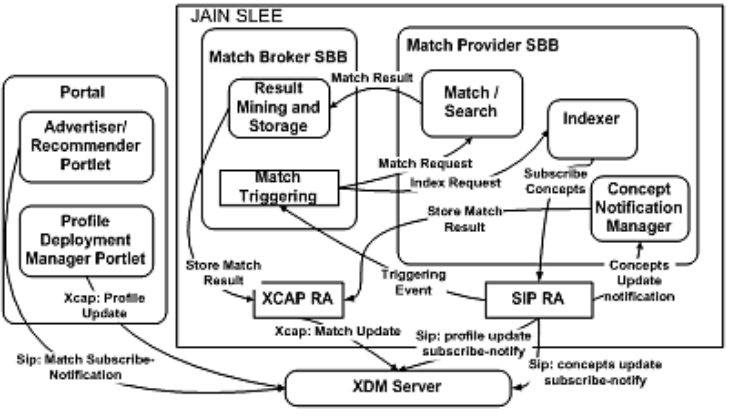

Fig. 5. Computational Model

3.1 OPUCE Platform

In the OPUCE project platform, as shown in figure 6, users are allowed to create services by composing already deployed base services. The Matcher is integrated by interacting with the Portal, and with the Service Advertiser and User Information Manager sub-systems as a sub-module. It is used to match services with newly deployed user profiles and vice versa.

As explained, in the future it will be used as a search engine, allowing a service creator to find the Base Services that he needs, by deploying a dummy profile that is never indexed and is only used for matching. In addition, it allows a user to find already deployed services that he needs, either using describing concepts or the service's description on a formal semantic language such as Web Service Description Language [17] [18]. The search will also be performed by deploying a profile, a *search profile* for this matter.

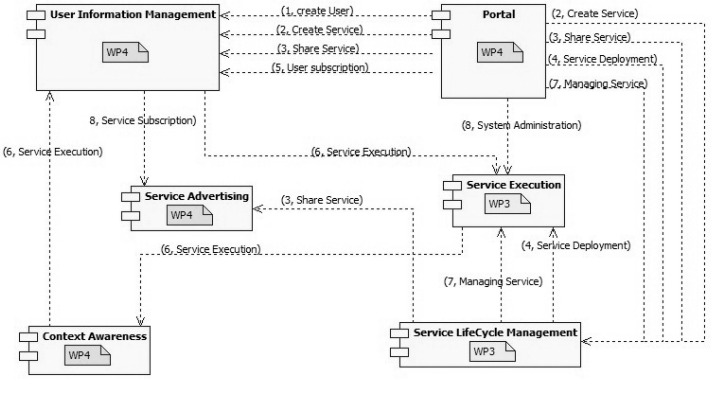

Fig. 6. OPUCE Architecture

3.2 Matcher at OMA IMS and World Wide Web

This platform can potentially fit as a functional element on the IMS / OMA managing the deployment of services, which are easily composed in this service oriented architecture, whose interfaces are internally based on SIP and XCAP, and externally is wrapped as a Web Service. This integration is detailed in figure 7.

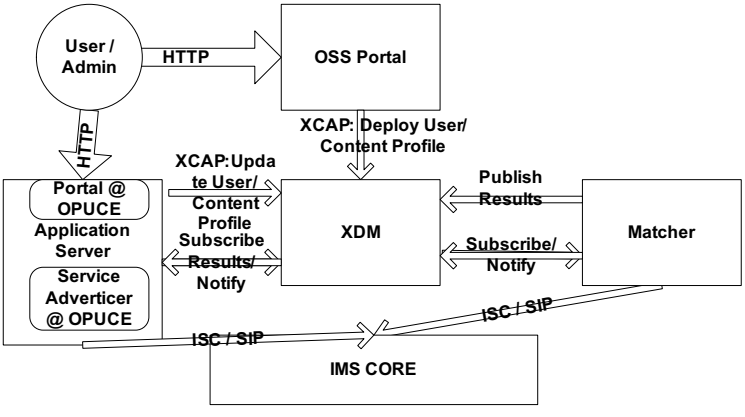

Fig. 7. IMS integration Architecture

Our platform can be used as a match/search engine inside content servers in the world wide web. Any online store or any other service provider that offers contents needs to make them available to search. Thus, a matcher tool that enables for advanced classification rather than mere tag classification becomes possible with our framework. Any content server, either it is an online store like `www.amazon.com` or a media server like `www.youtube.com`, requires that its users have profiles and they need to find contents matching them.

4 Conclusions

The current prototype still does not allow for complex semantic property composition for profile description, nor for composing queries with the *not* operator. It may deal with this complexity problem assuming a closed world. The semantic properties that can be used so far narrows down to rdf:about. It still does not cross recommend other profiles based on the recommendations done to similar profiles of the same type. We are yet to work on a Third Party Web Service interface wrapper *a la Parlay-X*. This will allow profile deployment both for indexing and match/query functionalities. The Web service has to be fully described with WSDL 2.0. Work must also be done on the integration with the XDM server in order to allow for the described mechanisms of complex profile owner subscribing to concepts. The usage of JSLEE technology must be revisited, since currently identified scenarios do not impose real time requirements.

References

1. Google search engine, http://www.google.com
2. Open mobile alliance, http://www.openmobilealliance.org/
3. Berners-Lee, T., Hendler, T., Lassila, O.: The semantic web. Scientific American (2001)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Dl-lite: Tractable description logics for ontologies. In: Veloso, M.M., Kambhampati, S. (eds.) Proceedings, The Twentieth National Conference on AI and the Seventeenth Innovative Applications of AI Conference, Pittsburgh, Pennsylvania, USA, pp. 602–607. MIT Press, Cambridge (2005)
5. Choi, N., Song, I.-Y., Han, H.: A survey on ontology mapping. W3C Recommendation (February 2004), http://www.w3.org/TR/owl-ref/
6. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: Deduction in concept languages: From subsumption to instance checking. Journal of Logic and Computation 4(4), 423–452 (1994)
7. Group, N.W.: The extensible markup language(xml)configuration access protocol(xcap). RFC 4825 (May 2007), http://tools.ietf.org/html/rfc4825
8. IBM. Ibm websphere group list server, version 6.1.1 - subscribe and notify
9. Jena.sourceforge. Jena 2 inference support, section 4. the owl reasoner
10. Klyne, G., Carroll, J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation (February 2004), http://www.w3.org/TR/rdf-concepts/
11. OMA. Oma Xml Document Management (xdm) v1.0.1
12. Ortiz, M., Calvanese, D., Eiter, T.: Characterizing data complexity for conjunctive query answering in expressive dl. In: The Twenty-First National Conference on AI and the Eighteenth Innovative Applications of AI Conference, Boston, Massachusetts, USA, July 16-20 (2006)
13. Partners. Open platform for user-centric service creation and execution
14. . N. C. S. C. C. . Sun Microsystems, Inc. Jain and java in communications
15. Thomas Kuhn, E.M.G., Olivier Thomann, I.O.L.: Abstract syntax tree
16. van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL web ontology language reference. W3C Recommendation (February 2004), http://www.w3.org/TR/owl-ref/
17. W3C. Web services description language (wsdl) version 2.0 part 1: Core language. W3C Working Draft (November 2003), http://www.w3.org/TR/2003/WD-wsdl20-20031110/
18. W3C. Web services description language (wsdl) version 2.0 part 0: Primer. W3C Working Draft (December 2004), http://www.w3.org/TR/wsdl20-primer