Construction d'application Mashup à base d'annotations sémantiques

Abdelhamid Malki, Sidi Mohammed Benslimane, Mimoun Malki

Département d'informatique
Université Djilali Liabes de Sidi Bel Abbes
Algérie

abdelhamid.malki@gmail,{Benslimane,malki}@univ-sba.dz

RÉSUMÉ. Les Mashups ont permis une avancée significative dans l'automatisation des interactions entre les applications et ressources Web. Notamment, la combinaison des APIs Web est considérée comme un point fort, qui permet de répondre à des besoins complexes en combinant les fonctionnalités et les données de plusieurs services au sein d'une seule application Mashup. L'automatisation du processus de construction de Mashup repose principalement sur la sémantisation des APIs Web afin de faciliter au développeur leur sélection et leur mise en correspondance. Dans ce papier, nous proposons SAWADL (Semantic Annotation for Web Application Description Language), une extension du langage WADL, pour la sémantisation des services Web REST. Nous introduisons une architecture de référence à cinq couches représentant les principaux blocs fonctionnels permettant d'annoter et de combiner les APIs Web, et par conséquence rendre plus agile et plus flexible le processus d'ingénierie des applications Mashup.

ABSTRACT. Mashups allowed a significant advance in the automation of interactions between applications and Web resources. In particular, the combination of Web APIs is seen as a strength, which can meet the complex needs by combining the functionality and data from multiple services within a single Mashup application. Automating the process of building Mashup based mainly on the Semantics Web APIs which facilitate to the developer their selection and matching. In this paper, we propose SAWADL (Semantic Annotation for Web Application Description Language), extension of the WADL language that allows the semantization of the REST Web Service. We introduce We introduce a reference architecture with five layers representing the main functional blocks for annotating and combining web APIs, and therefore make the engineering process of Mashup applications more agile and more flexible.

MOTS-CLÉ: Mashup Sémantique, Matching, API, SOAP, REST, SAWADL, SAWSDL.

KEYWORDS: Semantic Mashup, Matching, API, SOAP, REST, SAWADL, SAWSDL.

1. Introduction

En raison des carences en matière d'agilité et de flexibilité des systèmes d'information d'entreprise, une nouvelle approche appelée *Mashup* est née. Les Mashups sont des applications web développées par la combinaison: des données, des logiques métiers, et/ou des interfaces utilisateurs [8] des sources web publiées et réutilisées via des APIs. Ainsi, Les Mashups visent à réduire le coût et le temps de développement des applications web.

Malgré ces avantages, l'ingénierie des applications Mashups nécessite l'intervention du développeur qui a besoin non seulement des compétences en programmation mais aussi de comprendre la structure et la sémantique des APIs qu'il souhaite intégrer. Actuellement, plusieurs outils Mashup sont utilisés pour faciliter aux utilisateurs finaux la construction des applications Mashup (e.g. *IBM WebSphere*[1], *Yahoo-pipes*[2], etc.). Néanmoins l'intervention du développeur professionnel est nécessaire lorsque l'application Mashup est complexe, chose qui a poussé les chercheurs à trouver des solutions efficaces pour la création des Mashups de manière qu'un utilisateur final peut construire une application Mashup avec un outil lui garantisse la découverte, la sélection, et la superposition automatique ou dynamique des APIs en se basant sur l'approche sémantique, ce qu'on appelle *Mashup Sémantique*. Les Mashups sémantiques sont des Mashup dont les APIs combinés sont soutenus (ou annotés) par une couche sémantique qui permet de les sélectionner et les composer de manières automatique (non-ambigüe).

Nous proposons dans ce travail SAWADL (Semantic Annotation for Web Application Description Language), un nouveau langage pour la sémantisation des services web REST [1]. SAWADL utilise la description WADL[3] afin d'enrichir les APIs de type REST avec une couche sémantique qui permet la découverte et la superposition automatique des APIs afin de construire automatiquement des applications Mashup. SAWADL est plus flexible et adaptative vis-à-vis des autres approches de sémantisation tel que l'approche SAWSDL [2] qui sert à annoter la description WSDL[4] des services web SOAP[5] avec des concepts ontologiques.

Le reste du papier est organisé comme suit. La section 2 présente les travaux relatifs à la problématique de sémantisation des services web REST. Dans La section 3 nous présentons le langage SAWADL qui permet de sémantiser les APIs de type REST. Une démarche de construction de Mashup sémantique est décrite dans la section 4. Enfin nous concluons et donnons quelques perspectives dans la section 5.

[1]. http://www-01.ibm.com/software/webservers/
[2]. http://pipes.yahoo.com/pipes/
[3] http://www.w3.org/Submission/wadl/
[4] http://www.w3.org/TR/wsdl
[5] www.w3.org/TR/soap/

2. Travaux connexes

Les services Web permettent aux applications d'utiliser des fonctionnalités à distance et d'échanger des données en passant par des messages biens définis. Cela peut facilement être utilisé par une application Mashup comme une façon d'orchestrer différentes applications web. Par exemple, *AmazonWebService*[6] permet aux utilisateurs d'accéder à la plupart des fonctionnalités d'Amazon.com en utilisant des services web basés sur SOAP et REST. Dans le but de construire automatiquement des applications Mashup il est nécessaire de sémantiser ces APIs.

Pour les services web SOAP, il existe deux types d'approches de sémantisation. La première approche consiste à développer un langage complet qui décrit les services Web ainsi que leur sémantique dans un seul bloc (e.g. OWL-S[7], WSMO[8]). La deuxième approche consiste à annoter les langages existants avec de l'information sémantique. WSDL-S[9], et SAWSDL permettent d'annoter manuellement une description WSDL avec des éléments faisant référence à des ontologies.

Comme pour les services web SOAP, la sémantisation des services web REST peut être classifiée en deux approches. La première approche consiste à développer une ontologie qui décrit les services Web REST ainsi que leur sémantique dans un seul bloc. La deuxième approche consiste à annoter les langages existants avec de l'information sémantique. Dans ce que suit, nous présentons les principales approches existantes dans la littérature.

2.1 SOOWL-S (Social-oriented OWL-S) advertisements

L'approche SOOWL-S advertisements [6] propose une extension de l'ontologie OWL-S dans le but de sémantiser les différents types d'APIs (e.g. SOAP, REST, JS, RSS,..) utilisés dans la construction des applications Mashup. L'ontologie SOOWL-S permet juste d'annoter les paramètres d'entrées /sorties et les propriétés non-fonctionnelles d'un service Web. Ainsi, l'ontologie SOOWL-S permet la recherche et la sélection automatique des APIs, mais pas leur combinaison à cause de l'absence de l'extension du module *service-Model* de l'ontologie OWL-S.

2.2 SA-REST(semantic annotation for REST)

D'après Lathem et al. [4], la plupart des services web RESTful ont des pages HTML qui décrivent aux utilisateurs ce que le service fait et comment l'invoquer. C'est en quelque sorte l'équivalent d'un WSDL pour les services web RESTful, cela serait le moyen idéal pour ajouter les annotations sémantiques. Le problème est que HTML est conçu pour être lisible par l'homme et non pas par une machine.

[6] http://aws.amazon.com/ (02.01.2012).
[7] http://www.w3.org/Submission/OWL-S/
[8] http://www.wsmo.org/
[9] http://www.w3.org/Submission/WSDL-S/

Afin de résoudre ce problème, [4] ont utilisé le micro formats RDFa[10] qui permet d'intégrer des triplets RDF au-dessus de la description HTML afin d'ajouter la sémantique au service REST et le rendre interprétable par la machine. Dans cette approche, les éléments annotés dans les services Web RESTful sont les sorties, les entrées, les opérations, ainsi que le type de la requête qui peuvent invoquer le service.

2.3 SWEET (Semantic Web Services E-diting Too)

La plupart des services web REST sont décrits en HTML; par conséquent, l'absence d'une description lisible et interprétable par la machine rende difficile voire impossible d'ajouter des annotations sémantiques au-dessus des services web REST afin de fournir un certain niveau d'automatisation. Ainsi, Maleshkova et al. [5] ont proposé SWEET, une approche intégrée pour décrire formellement la sémantique des services web RESTful. Ils commencent par la création d'une description lisible par la machine pour les services web REST en utilisant hRESTS (HTML pour RESTful Services) [3]. Par la suite hRESTS sera complété par MicroWSMO[11] pour l'annotation sémantique des services web REST.

La Table 1 illustre une comparaison entre les différentes approches de services web REST sémantique.

TABLE I. *Comparaison entre les différentes approches*

	SOOWL-S	**SA-REST**	**SWEET**
Type de sémantisation	Ontologie de service	Annotation	Annotation
Publication de service	+	-	+/-
Découverte de service	+	+	+
Combinaison de service	-	+	+
Description annotée	Entrées/sorties de service	HTML	HREST
Ontologie supportée	OWL	Toutes	Toutes
API sémantisée	SOAP, REST, RSS, JS	REST	REST

3. SAWADL

Dans cette partie nous proposons une approche qui permet de sémantiser les services WEB RESTful (les plus utilisé dans les applications Mashups) afin de renforcer la sélection et la superposition de ces services dans les applications Mashups.

SAWADL, l'extension du langage WADL que nous proposons fait partie des approches qui permettent d'ajouter des annotations sémantique au-dessus de la description du service. Comme c'est invoqué dans section précédente, la plupart des approches sont basées sur une annotation sémantique au-dessus d'une description fondée sur HTML, ce qui empêche l'interopérabilité entre les services web REST sémantisés. SAWADL ne spécifie pas un langage pour la représentation des modèles sémantiques. Il fournit plutôt des mécanismes pour référencer des concepts de mo-

[10] http://www.w3.org/TR/rdfa-syntax/
[11] http://www.wsmo.org/TR/d38/v0.1/

dèles définis à l'extérieure du document WADL. Les méthodes d'annotation dans SAWADL se résument en deux mécanismes: `modelReference et SchemaMapping`. Pour éviter toute ambiguïté, ces deux mécanismes sont précédés par le préfixe par `sawadl`.

L'attribut `modelReference` permet d'associer un composant WADL à un concept d'un modèle sémantique (e.g. ontologie). Les éléments annotés d'un service Web REST décrit par la description WADL sont les méthodes (`<method>`) et les paramètres d'entrées/sorties (`<param>`) du service. Le concept sémantique associé aux éléments de WADL par le biais de l'attribut `modelReference` est représenté par zéro ou plusieurs URLs séparées par des espaces et qui font références à des concepts ontologiques.

Le mécanisme de `schemaMapping` est réalisé par le biais des attributs `liftingSchemaMapping` et `loweringSchemaMapping`. Ces attributs permettent de spécifier les mappings entre les données sémantiques et les éléments WADL. L'attribut `loweringSchemaMapping` est employé lorsqu'un élément annoté dans la description WADL correspond à plusieurs concepts ontologiques. Les URIs de l'attribut `loweringSchemaMapping` pointent vers des fichiers contenant des requêtes SPARQL[12] ou des transformations XSLT[13]. L'attribut `liftingSchemaMapping` est utilisé lorsque plusieurs éléments annotés dans la description WADL correspondent à un seul concept ontologique. Les URIs de l'attribut `loweringSchemaMapping` pointent vers des fichiers contenant des requêtes XSLT ou XQuery[14].

3.1. Annotation des méthodes

SAWADL fournit des mécanismes pour l'annotation des méthodes dans un document WADL. Pour illustrer ces mécanismes, nous utilisons une ontologie du domaine de tourisme `TravelOnto` (que nous avons implémenté en OWL) pour annoter l'opération `ReserverVol` d'API *Vol*. Bien que traditionnellement les entrées et les sorties d'une opération fournissent d'une manière intuitive la sémantique d'une opération, une annotation sémantique simple peut s'avérer utile. Ainsi nous allons annoter l'opération `ReserverVol` en l'associant par le biais de l'attribut `modelReference` au concept `BookFlight` de l'ontologie `TravelOnto` (Figure.1).

```
1 <resources base="http://localhost/ApiRest/Vol">
2 <method id="ReserverVol " name="GET"
3 sawadl:modelReference=TravelOnto#BookFlight">
4 <request>
5 <param name="Nom" type="xsd:string"/>
6 <param name="Prénom" type="xsd:string"/>
7 <param name="Age" type="xsd:int"/>……
```

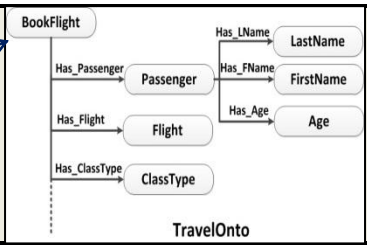

Figure 1. *Annotation des méthodes avec SAWADL*

[12] http://www.w3.org/TR/rdf-sparql-query/
[13] http://www.w3.org/TR/xslt
[14] http://www.w3.org/TR/xquery

3.2. Annotation des entrées sorties

Cette section se focalise sur la façon dont est faite l'annotation des entrées et des sorties dans SAWADL. En effet, l'annotation d'une entrée ou une sortie peut se faire de façons différentes :

3.2.1. Annotation de niveau interne

L'annotation au niveau interne d'un fichier WADL consiste à associer à chaque paramètre d'entrée/sortie `<param>` d'une méthode un concept dans une ontologie. Cela suppose que pour chaque paramètre d'entrée/sortie d'une méthode il existe un concept correspondant dans l'ontologie. Par exemple, l'entrée de l'opération `ReserverVol` est composée de nom, prénom et l'âge du passager, et le numéro et la classe de vol. Nous supposons que pour chaque entrée, il existe un concept qui lui correspond dans l'ontologie OWL `TravelOnto` (voir Figure 2). Dans le cas où il n'y a pas de correspondance, la sémantique des paramètres d'entrées/sorties reste non spécifiée.

```
1 <resources base=" http://localhost/ApiRest/Vol ">
2 <method id="ReserverVol" name="GET">
3 <request>
4 <param name="Nom"  type="xsd:string" sawadl:modelReference="TravelOnto#LastName"/>
5 <param name="Prénom" type="xsd:string" sawadl:modelReference="TravelOnto#FirstName"/>
6 <param name="Age"  type="xsd:int" sawadl:modelReference="TravelOnto#Age"/>
7 <param name="NVol" type="xsd:string" sawadl:modelReference=" TravelOnto#Flight" />
8 <param name="Classe" type="xsd:String" sawadl:modelReference="TravelOnto#ClassType"/>
9 </request>…
```

Figure 2. *Annotation au niveau interne*

3.2.2. Annotation de niveau externe

Au niveau externe, les entrées/sorties d'une méthode sont annotées globalement via la balise `<request>`. Cependant, il faut créer des *mapping* qui permettent de spécifier les schémas de transformation nécessaires entre les paramètres de la méthode et les concepts d'une ontologie.

A titre d'illustration, nous prenons l'exemple de l'API `Carte Bancaire` définit en WADL et annotée par l'ontologie OWL `TravelOnto` (voir Figure 4). Dans cette ontologie il n'y a pas de correspondance individuelle pour les deux attributs `Nom` et `Prénom`. Cependant, le concept `Owner` de l'ontologie correspond à la fusion de ces deux attributs. Afin d'établir la correspondance entre les entrées de l'API `Carte Bancaire` et le concept `CreditCard` il faut au préalable les associer en utilisant `sawadl:modleReference` et ensuite définir un schéma de transformation à l'aide d'une feuille de style XSL via l'attribut `sawadl:liftingSchemaMapping` (voir Figure 3).

```
<xsl:transform version="2.0"
   xmlns:Travel=http://localhost/ApiRest/Billet#  xmlns:TravelOnto="http://localhost/TavelOnto#>
   <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes"/>
   <xsl:template match="/"><rdf:RDF>
    <TravelOnto:CreditCard>
     <hasOwner  rdf:resource="#Owner">
       <xsl:value-of select="concat(Travel:./param[@name='Nom'],Travel:./param[@name='prénom'])"/>
     </hasOwner>
     <hasCardNumber rdf:datatype="xs:Int"><xsl:value-of select="Travel:./param[@name='Numéro']">
     </hasCardNumber>
     <hasType rdf:datatype="xs:string"> <xsl:value-of select="Travel:./param[@name='Type']">
     </hasType>
     <hasExpritionDate rdf:datatype="xs:Date"> <xsl:value-of select="Travel:./param[@name='DateExpri']">
     </hasExpritiondate>
   </TravelOnto:CreditCard></rdf:RDF>
  </xsl:template>
</xsl:transform>
```

Figure 3. *Feuille de style XSL via l'attribut* `sawadl:liftingSchemaMapping`

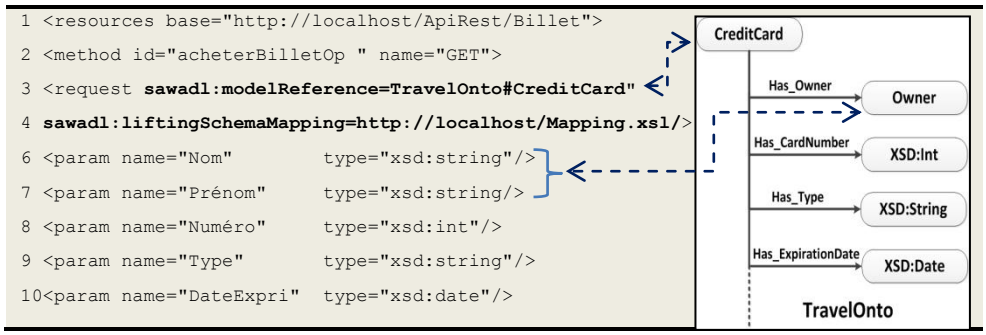

Figure 4. *Annotation externe*

4. Construction de Mashup sémantique avec SAWADL

La création automatique des applications Mashup nécessitera forcement une couche sémantique au-dessus des APIs (service web). Comme la composition dynamique des services web classiques, les Mashup sémantique permettent un développement plus rapide et une composition transparente à l'utilisateur. Mais contrairement à celle des services web classiques les Mashup se composent d'APIs de différentes natures ce qui rend leur processus de combinaison difficile.

La Figure 5 montre une architecture de référence pour un Mashup sémantique. Cette architecture se compose de cinq couches représentant les principaux blocs fonctionnels pour la génération automatique des Mashup. Une ontologie de domaine est utilisée pour enrichir le processus d'ingénierie par une spécification sémantique permettant l'annotation, la découverte et la combinaison automatique des APIs participant à la génération de l'application Mashup.

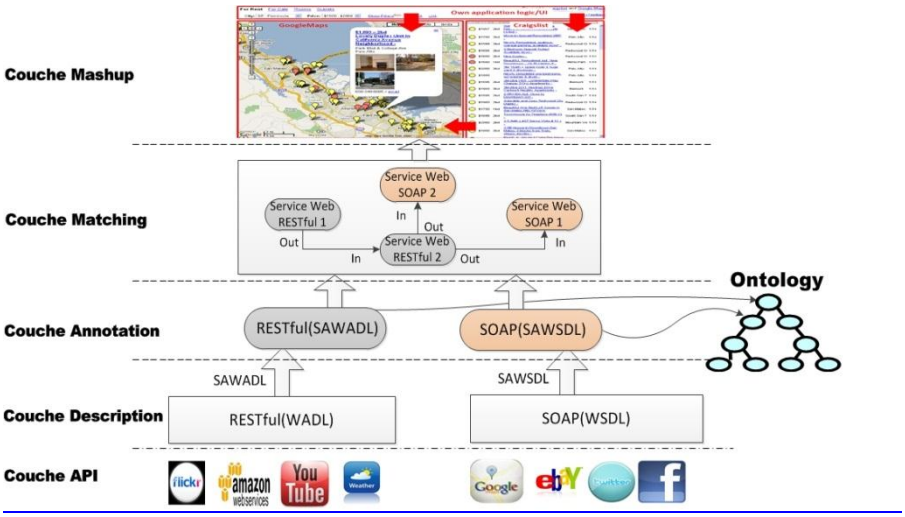

Figure 5. *Architecture de référence d'un Mashup sémantique*

A. La couche API

A ce niveau plusieurs types d'APIs sont concernés. En particulier les APIs à base de SOAP et RESTful qui sont les sont les plus utilisés dans l'ingénierie des applications Mashups.

B. La couche description

Au niveau de cette couche, les langages WADL et WSDL sont utilisés respectivement pour la description des APIs REST et SOAP.

C. La couche annotation

En plus du langage SAWADL que nous proposons dans ce papier. Plusieurs langages d'annotation de services web sont considérés à ce niveau. En particulier, SAWSDL qui est utilisée pour sémantiser les services web SOAP par l'annotation des entrées/sorties d'un fichier WSDL avec des concepts ontologiques. Cette couche va servir dans la construction automatique des Mashups en permettant la découverte, la sélection et la combinaison de manière non-ambigüe des différentes APIs.

D. La couche Matching (Correspondance)

L'hétérogénéité entre les annotations décrites par SAWSDL et celles décrites par SAWADL est résolue par les quatre règles suivantes :

- ***Règles 1.*** Une méthode décrite par la balise <*method*> d'une ressource ou une sous-ressource < *resource*> d'un fichier SAWADL correspond à une opération décrite par la balise <*operation*> d'un fichier SAWSDL.
- ***Règles 2.*** Une entrée décrite par la balise <*param*> de l'élément <*request*> d'une méthode d'un fichier SAWADL correspond à une entrée décrite par la balise <*element*> d'un <*complexType*> d'une opération décrite dans un fichier SAWSDL.

- ***Règles 3.*** Un Output décrit par la balise <*response*> d'une méthode d'un fichier SAWADL correspond à une sortie décrite dans le XML schéma d'un service web par la balise <element> d'un <complexType> d'un output d'une opération décrite dans un fichier SAWSDL.
- ***Règles 4.*** Les attributs "*sawadl: modelReference", "sawadl: liftingSchemaMapping", "sawadl:loweringSchemaMappin"* d'un fichier SAWADL correspondent respectivement aux attributs *"sawsdl: modelReference", "sawsdl:liftingSchemaMapping", "sawsdl: loweringSchemaMappin"* d'un fichier SAWSDL.

Les correspondances entre les APIs sont établies en se basant sur la similarité sémantique décrite dans [7]. Cette similarité permet de calculer une distance entre les concepts ontologiques annotant les entrées/sorties des APIs.

La valeur de Matching entre deux services peut être calculée en utilisant la formule suivante :

$$Match(A_{in}, A_{out}) = \sum (1 - dist(P_{in}, P_{out})\ ) / (n_{in} + n_{out})$$

Où n_{in} et est le nombre des paramètres d'entrés de l'API A_{in}, n_{out} est le nombre des paramètres de sorties de l'API A_{out}, et $dist(P_{in}, P_{out})$ est la distance sémantique entre les paramètres d'entrées et sorties des APIs $A_{in}\ et\ A_{out}$ [7] .

E. La couche Mashup

Au niveau de cette couche une application Mashup est réellement créée en se basant sur les résultats obtenus par la couche Matching. La couche Mashup intègre les APIs qui ont une valeur de Matching supérieur ou égale à un seuil prédéfinit par des experts de domaine. La combinaison des APIs peut être faite en utilisant différentes technologies (e.g. Ajax, Php, Jsp, etc.). Le résultat de la combinaison est validé par l'utilisateur de l'application Mashup.

5. Conclusion

Les Mashups sont des applications web développées par la combinaison des données, des processus métiers, et/ou des interfaces utilisateurs des sources Web publiées et réutilisées via des APIs. Ainsi, les Mashups visent à réduire le coût et le temps de développement des applications Web. Cependant, afin de remédier aux carences des langages et protocoles actuels mis en place par la communauté informatique, les travaux liés à l'ingénierie des applications Mashups se sont particulièrement orientés vers le niveau sémantique. L'objectif recherché à travers l'utilisation de la sémantique est de permettre aux machines d'interpréter les données traitées et les APIs manipulés pour la construction automatique des applications Mashup.

De nombreux langages ont été proposés pour l'annotation sémantique des APIs de type REST (voir section 2). Cependant, ces approches nécessitent une page web HTML qui décrit l'API et qui sera transformée par la suite en une description interprétable par la machine afin d'ajouter des annotations sémantique. Chose qui n'est

pas toujours vrais et qui rend la tâche plus difficile surtout si l'API REST ne dispose pas d'une page web HTML qui la décrit. C'est dans le but de répondre à ces problématiques que nous avons mené nos recherches.

Dans ce papier nous avons proposé SAWADL, un langage d'annotation sémantique pour les services Web REST. Notre langage SAWADL fait partie des approches qui permettent d'ajouter des annotations sémantique au-dessus de la description du service. Contrairement aux approches qui annotent au-dessus d'une description HTML, nous utilisons la description WADL qui est utilisée pour décrire syntaxiquement les services web REST. La sémantisation des APIs ne suffit pas pour concevoir et mettre en œuvre un Mashup automatique. Ainsi un processus de Matching est nécessaire pour trouver des correspondances entre les différents APIs, afin de découvrir de manière automatique les composants Mashables suivant les besoins exprimés par les utilisateurs.

6.References

[1] R.Fielding, Architectural Styles and the Design of Network-based Software Architectures, PhD thesis, University of California, 2000.

[2] J.Kopecký, T.Vitvar, C.Bournez, J.Farrell: SAWSDL: Semantic Annotations for WSDL and XML Schema, IEEE Internet Computing, vol. 11, no. 6, pp. 60–70, November-December 2007.

[3] J. Kopecky , K. Gomadam, T.Vitvar: hRESTS: an HTML Microformat for Describing RESTful Web Services , Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI-08), 2008.

[4] J.Lathem, K.Gomadam, P. Sheth; SA-REST and (S)mashups Adding Semantics to RESTful Services , Proceedings of the First IEEE International Conference on Semantic Computing (ICSC 2007), September 17-19, 2007, Irvine, California, USA. IEEE Computer Society 2007.

[5] M.Maleshkova, C.Pedrinaci, J.Domingue, Supporting the Creation of Semantic RESTful Service Descriptions, 2009, In: 8th International Semantic Web Conference (ISWC 2009), 25-29 Oct 2009, Washington D.C., USA.

[6] G. Meditskos, N. Bassiliades , A combinatory framework of Web 2.0 mashup tools, OWL-S and UDDI, Expert Systems with Applications, vol. 38, no. 6, pp. 6657–6668,June 2011.

[7]H.H. Ngu Anne, P. Carlson Michael, Z Quan Sheng. Semantic-based Mashup of Composite Applications, IEEE Internet Computing, vol. 3, no. 1, pp. 2–15,January-March 2010.

[8] J.Yu, B.Benatallah, F.Casati, F.Daniel. Understanding Mashup Development and its Differences with Traditional Integration, , IEEE Internet Computing, vol. 12, no. 5, pp. 44–52,September-October 2008.