A new mediator-based architecture for the dynamic service composition

Tuo Zhang, Ken Chen
L2TI, Institute Galilée
University Paris 13
Paris, France
tuo.zhang@univ-paris13.fr,
ken.chen@univ-paris13.fr

Jiwen Yan, Antoine Bouhier
L2T1, Institute Galilée
University Paris 13
Paris, France
yan.jiwenyi.jiwen@gmail.com,
antoine.bouhier@gmail.com

***Abstract*— Nowadays, the effective and adaptive dynamic Web service composition is a major challenge for a real success of Web services. In fact, the heterogeneity of the environment and autonomy of web services make it difficult to compose web services dynamically from various service providers. This problem is also accentuated by the increase of the user mobility. By analyzing the actual technology and its evolution, we propose in this article a mediator-based architecture that allows users to dynamically compose the ubiquitous web services. Furthermore, the approach we have adopted also facilitates the semantic web. A model which is implemented in .NET has validated the feasibility of our proposal.**

Keywords-Web service composition; context adaptation; mediator; semantic web; user-centric.

I. INTRODUCTION

With the rapid development of the web technology, the Web Service (WS) has attracted great attention in the industry domain. The WS, with its full autonomy and loose-couple interface, provides an ideal integrate approach for the implementation of the application deployed on the Web in various domains.

With the rise of the applications based on the Internet, as well as the B2B/B2C and the development of Cloud Services and Internet of Things (IoT), service integration and performance requirements continue to increase. It is difficult to meet the user's needs through a single specific service. So, the dynamic service composition attracted the attention of academics and business in the past few years. However, there are several problems to solve in the process of composing dynamic services. First of all, we cannot easily integrate the WSs from the different suppliers because of the heterogeneity of the environment and the autonomy of WS. Secondly, due to the dynamic nature of the operation and process treatment, coding in a fixed form is no longer applicable. Thirdly, because of the user mobility and their diverse needs, we need a mechanism to dynamically discover, select and combine the WSs in the inter-organization and cross-platform.

With the development of the Cloud Computing and the occurrence of the increasingly diverse WSs, as well as the democratization of mobile terminal (Smartphones, Tablet), more and more users would like to perform a sequence of operation automatically according to their logic which refers to use a series of individual existing WS.

Therefore, it is appropriate for us to conceive architecture to provide the service composition such a benefit with the development of SOA. One of the key issues is the semantic web. It allows not only to solve the problem of mismatching between the web services, but also to automatically identify the functional equivalent service entities. The latter is very useful to answer another key question that is the adaptation to the context (or ubiquity), which is to be able to replace a service entity by another functionally equivalent one. We believe that the mediator-based approach that we propose could be a response to these two key questions.

In this paper, our work is in the context of NGN/NGS (Next Generation Network/ Next Generation Service) which refers to the paradigms—heterogeneity, mobility and user-centric. Based on analyzing current language, technology and the method of the service composition, we propose a mediator-based architecture to achieve dynamic service composition. We present the overall vision and we discuss the technical solutions for the key component within it. We illustrate the feasibility of our proposal through a model that we have achieved and we present an application of our architecture through a use case.

In our model, the mediator has a Knowledge Base of WSs, each of them is assigned with a descriptor, a kind of meta-model that contains various information, including the inputs/outputs and constraints, the syntactic and semantic (for the semantic web extension), and the location of available entities to provide service (allows context adaptation) as well as the contextual information. A natural extension is the creation of meta-WS, which can be integrated into the above knowledge base exactly in the same manner as a real basic WS. These meta-WSs are a kind of "proxy". Thus, our WS-mediator also allows the

 ISBN: 978-1-61208-232-5

creation of the true "intermediaries" to which we refer as a semantic extension. In particular, a WS-mediator can be used to select, at run time for instance, the best WS according to the using context. Exploring the knowledge on the existing WS for the purpose of semantic web can also create such a WS-mediator. On the point related to the context adaptation, our work is conducted partially within the French ANR/VERSO/UBIS project, which proposes a general architecture providing ubiquitous services that include identifying, for each service (within the meaning of abstract term), functionally equivalent services entities. Our work has been in effect for the partial framework ANR / UBIS.

This article is organized as follows. In Section 2, we present the related work about the current technology and solutions of the service composition. In Section 3, we conceive our proposal -- a mediator-based architecture and its main features. Section 4 shows our model which is implemented based on the .NET 4.0. An application on a scenario is illustrated in Section 5. Finally, conclusion and the future work are presented in Section 6.

II. RELATED WORK

Different views and focuses on the approaches for the Web service composition have been suggested in the literature during the last years. All of these try to provide languages, methods and models in order to propose efficient solutions for this problem.

Generally speaking, there are three types of language for the service composition. First of all, there is the process-oriented description language. Such as the BPEL4WS [1] (Business Process Execution Language for Web Service) and WS-CDL [2] (Web Services Choreography Description Language), BPEL4WS is largely used in the industrial domain. It devises a business process into an abstract and executable process. In addition, it defines the model and description language of the business process behavior. Thus, it greatly facilitates the process description and execution. WS-CDL describes peer-to-peer collaborations of participants by defining, from a global viewpoint, their common and complementary observable behavior, where ordered message exchanges result in accomplishing a common business goal. Both languages are XML-based, so they are usually applied to describe the Orchestration and Choreography [3]. Secondly, AXML [4] is a data-oriented language that is used in some specific areas such as astronomy or meteorology in order to handle the heterogeneous and massive data sources. The third type is the semantic-oriented language such as OWL-S [5] (Semantic Markup for Web Service) and SAWSDL (Semantic Annotation Web Service Description Language) [6]. By adding the semantic information into the description of Web service, it is possible to make the data and functionality machine-understood in the life cycle of a WS, and then generate a dynamic process of composing various WSs automatically.

With regard to the existing methods and tools for service composition, the WSMF (Web Service Modeling Framework) coined the term data mediation in the WSs context, the aim is to strengthen the semantic feature in order to automatically discover, compose and execute the services. It defines preconditions and effects and can be used for semantic annotation of WSDL with WSDL-S. However, there is no mechanism to describe Choreography or Orchestration in the WSMF, which seems an incomplete mix of semantic and syntactic. The WSMX (Web Services Modeling Execution Environment) is the reference implementation of WSMO [7][8] (Web Service Modeling Ontology). The principle of WSMF design is: a) strict decoupling of various components of the web service; b) use of the mediator to coordinate the various components. Thus, WSMF defines four elements, i.e. Ontology, Goal, Web Service and Mediator. WSMX creates an environment for WSMF and then increase business process automation in a very flexible manner while providing scalable integration solutions. The METEOR-S [9] project at the LSDIS Lab, University of Georgia, aims to extend the standards (SOAP, WSDL, UDDI) with Semantic Web technologies to achieve greater dynamism and scalability. They endeavor to define and support the complete lifecycle of Semantic Web processes [10]. In SWORD [11], a service is represented by a rule that expresses certain given inputs, and the service is capable of producing particular outputs. A rule-based expert system is then used to automatically determine whether a desired composite service can be realized using existing services. SWORD does not require (but could benefit from) wider deployment of emerging service-description standards such as WSDL, SOAP, RDF and DAML. Comparing to the traditional Petri Net based framework for Web services composition, the model proposed in [20] makes use of a kind of high level Petri nets called G-Nets instead of elementary ones. Basic and advanced constructs which are supported by the proposed algebra are syntactically and semantically defined. Considering the no-functional aspect, [23] provides a model which meets the QoS requirements of service consumers, guarantees the availability of Web Services composition and maximizes service providers' benefit. Meanwhile, [24] takes the QoS, user preference and the service relationships into account and proposes a method based Viterbi algorithm to reason out the global optimal solution of web composition service.

From the perspective of realization, we can define the service composition as manual, semi-automatic and automatic. The manual service composition [12] demands the user to clarify the process through GUI or a text editor, and then submit such process to the execution engine. For example, BPWS4J provides an eclipse plug-in which use WSDL and BPEL to describe the executable process. A model ontology tool named Protégé [13] can work with OWL-S API [14]. It can produce the OWL through WSDL, and then create the service composition with the service element manually selected by the user. The automatic or semi-automatic technology use the artificial intelligence (or AI planning) during the composition process. SWORD [11] adopting the service description based on rules, composes

 ISBN: 978-1-61208-232-5

services across details of the initial and final states, but it asks the user to specify the states of the service and lacks the part of service discovery. [21] propose a set of heuristics to effectively prune a large number of candidate abstract services and a novel approach to fully automate the generation of abstract services from a service community that consists of a set of functionally similar services. However, it lacks the human intervention. The IEEE Next-Generation Service Overlay Networks (NGSON) working group is focusing on the integration. The NGSON architecture was proposed [15] according to the NGSON concept with its extension for service composition. Sivasubramanian et al. [16] proposes some criteria to identify the levels of dynamism and automation in service compositions. Moreover, the NGSON group proposes a strategy where different techniques can be used to make compositions more automatic and dynamic with a model driven approach. However, a problem of such approach exists where, although the method can generate the process model automatically, there is a lack of the interactions between the designers and the process. This means it cannot accept the designer's decision as auxiliary information to generate next flow path during the composition process at runtime, because only the syntactic binding exists. Radiant [17] is an eclipse plug-in graphical tool that enables one to annotate existing Web service descriptions with Ontologies to create SAWSDL files [18]. Besides, there is also the "mash up" type composition approach, which offers a better legerity. For example, APIhut [19] builds a nice ecosystem in which one can reuse Web APIs, but advanced capabilities must also be developed in order to lead to dynamic configuration and composition for complex services. Moreover, following the survey and observation under the traditional composition context, Syu et al. [22] suggests two approach patterns and point out possible future challenges as well as directions, to the influence of the mature of mobile devices and environment.

Taking into account the current methods and the technologies of the service composition, we can add semantic information hierarchically in our mediator to realize our architecture. Concerning the specification and implementation of process, we adopt a process-oriented approach to achieve the control and execution for the service composition. In general, our architecture was integrated by several elements proposed in the existing solutions, for example the knowledge base of WSMF, the semantic annotation of METEOR-S, the matching of input/output in SWORD.

III. PROPOSITION: ARCHITECTURE AND FUNCTIONALITIES

We present hereafter the main architecture of our mediator and the functional entities within our approach.

The user begins by choosing the web services that he wants according to his logic. The WSs are registered in the Knowledge base with all the information of each WS element, and in particular, a model for each WS. Furthermore, the functional semantics, domain ontology and parameter ontology based Web service description methods can be used for WS composition that allow the semantic web. The composition is done, in our current prototype, through a Graphic User Interface, script-based extension will be added in the future. In this way, we get a composed execution entity that is totally autonomous. As the composition is based on the model of the WS, the Mediator-based system monitors Web Services at different locations in the Internet and dynamically assesses their dependability.

Figure 1 shows our proposal, initially, our mediator provides the basic function of data mediation for the service composition. The modules within the mediator work with the human intervention, the Process model of service composition can send its transaction taking into consideration of QoS, certain restraint and rules [23] as well as the models and algorithms [24]. In addition, the mediator adopts a knowledge base with WS available that can be dynamically composed. This database allows our architecture to provide both the extension to the Semantic Web and the ability of the context adaptation (ubiquity). Indeed, by identifying the WS (identified in the database) which are best suited to the user's semantic logic, the mediator is able to provide an appropriate service composition. On the other hand, we can also identify the service entity suited to the service (abstract) depending on the user preference through the same logic. This can be coupled with the service discovery that is dynamically fed by the knowledge base. The mediator can use the service entity which is best suited to the location of the user and ensure the context adaptation.

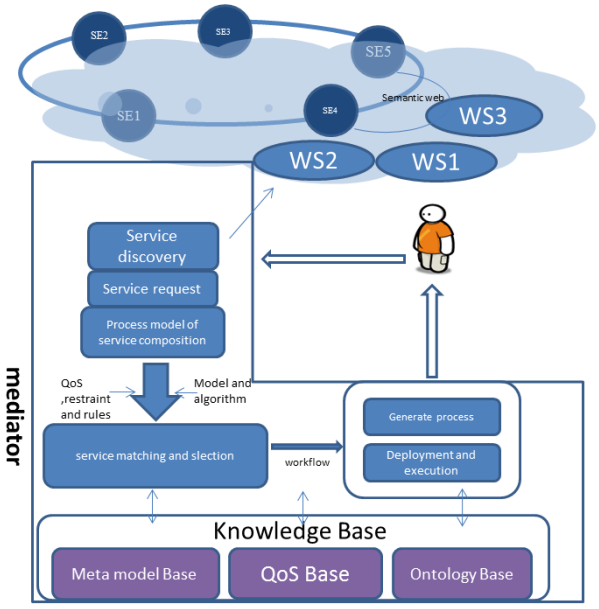

Figure 1. A mediator-based architecture

Our mediator can also provide the mapping between elements of SAWSDL. To illustrate various mapping representation options, we can use SPARQL for

representing mappings through the ontology knowledge base. The reasoning ability of ontology can help to resolve the substitution operation for the mismatching problem of web services so that we have a context adaptation with the semantic extension. The semantic extension will be integrated in accordance with certain rules for effective bonding to form a new web service composition. If there is no single service that could meet the user requirements, we can proceed by deductions and the dynamical combinations of semantic, based on the self-descriptions and the marks on the OWL-S of either the functional or the non-functional requirement among the known web services. Therefore, our approach enables us to access a loose-coupled way for the WS-mediator both on syntax and semantic. The implementation of the semantic extension (which will be based on existing tools [8] [11] [18]) is out of the scope of this paper.

A. *Knowledge base*

We get inspiration from storage of knowledge in WSMF and storage of XML in METEOR-S, and then we define a Knowledge Base which identifies the web services known to the mediator. The purpose of the Knowledge Base is to gather all the information (URI, operation, input / output from WSDL) of Web Services. The Knowledge Base is made manually (offline) in the current phase of our work. We define three sub-bases in our Knowledge Base:

- Meta model Base, which identifies the models of processes. For the common used services, a synoptic model can be generated by the service provider.

- QoS Base, which is used to store the QoS information. The mediator may find the semantic description of QoS based on WSDL during the period of discovery and matching. In the process of matching and optimization, QoS performance evaluation may be associated with the operation parameters during execution, so it is useful to save them to feedback. For example, considering the reliability of the service, we should refer to the success rate of execution because it is not comprehensive only depending on the service provider.

- Ontology Base, which is to store and manage the ontology service description and the semantic annotation, the upper level ontology and the domain specific ontology.

B. *Discovery, matching and selection*

Service discovery consists of two parts: a) the semantic discovery of each service request to create a process; b) the semantic discovery of service registered in UDDI that contains semantic information. The discovery of the semantics process, coupled with that of WS identified in Knowledge base, allows the identification of WS which may be involved in the process depending on their semantics. If we want to compose two web services, we will begin with choosing the services that we want to use, testing whether the services are available by the QoS restraint and rules, and checking the inputs and outputs of them. Both the syntactical and semantic heterogeneity may exist in the input and output messages that are exchanged between WSs. Once the services are ready to be used, the binding will be implied in the invocation of the composed services, and then we have the service selection done.

C. *Generation and Execution*

The execution entity receives the process model provided by the composition module. By setting the WSDL files and workflows, the output is an XML file that contains the entire process / composition. The execution entity receives the composition scheme provided by the composition entity. The mediator then generates an execution entity that is completely autonomous and re-usable.

IV. IMPLEMENTTATION

Based on SOA and WSC technologies, we have implemented our approaches and provided a comprehensive tool. The tool suite accepts WSs described using standard language such as WSDL as well as SAWSDL, which can provide us a semantic extension. The description of the process is done through an intuitive graphical interface to specify the user's logic, requirements and goals. From this description (which may be also provided as script), the component "mediator" of the system composes the services identified in the BDC, and then generates the composition as an execution entity that is available to the user.

We choose to implement the tool using web service developed in WCF (Windows Communication Foundation) of the Framework .NET 4.0. Because WCF supports not only SOAP message, but also it can be configured to support standard XML data that is not wrapped in SOAP, or can even be used to support other formats. This yields opportunities for evolutions such as the integration of the RESTful service.

We build the WCF based on three elements:

- Address: the address that the user must connect to use the service.
- Binding: the protocol to be used by the user to communicate with the service.
- Contract: the information exchanged between the mediator and the user so that he knows how to use the service.

The tool suite is an integrated developing environment for the process designers to

- Import candidate WSs and their description files
- Specify process hierarchies, initial state and goals

 ISBN: 978-1-61208-232-5

- Generate the plan and convert the plan into the corresponding execution entity.

In order to realize the three points mentioned above, we define and create the service contract and its parser module to provide the output of the previous WSs exactly match the required input of the successive WS. The Service contract is defined

- To be exchanged between the mediator and user
- To allow the user to know what are the methods proposed by the service and how to use them.

The development of the Service Contract is performed through the 3 following metadata:

- <ServiceContract>: This metadata is used to define a class. It serves to indicate the class or an interface is a Service Contract.
- <OperationContract>: This metadata is attached to the methods that we want to expose through WCF service. Thus, it is technically possible to expose certain methods of a class to the user.
- <DataMember>: This attribute is placed before the properties of classes to define objects that are then going to exchange the parameters with the service.

For example, we want to compose the WSs Multiplication, Subtraction and Addition (see Figure 2)

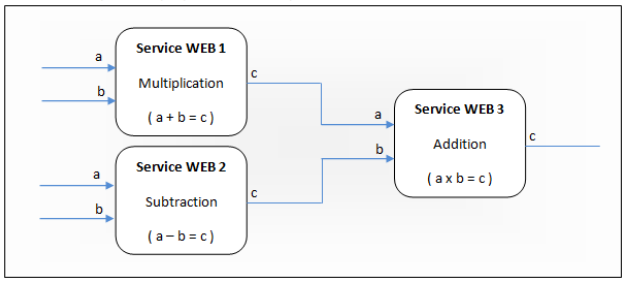

Figure 2. Example for the service composition

The service contract outlines the information that describes the service delivery. It defines a mechanism for the service orchestration between the service elements. It defines in particular the interface specification and describes the service logic and service purpose as to implement the process information about the service elements to supply a more efficient treatment. The service contract focuses on the organization and parsing of input/output data treatment as well as the QoS (if needed). Figure 3 provides the details of the service contract with schema XSD (XML Schema Document)

```
<?xml version="1.0" encoding="utf-8" ?>
<Contrat xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="Structure_Contrat.xsd">
  <Services>
    <Service>
      <ServiceNom>Addition</ServiceNom>
      <Inputs>
        <Input>
          <InputNom>a</InputNom>
          <InputType>int</InputType>
        </Input>
        <Input>
          <InputNom>b</InputNom>
          <InputType>int</InputType>
        </Input>
      </Inputs>
      <Outputs>
        <Output>
          <OutputNom>RéponseAddition</OutputNom>
          <OutputType>int</OutputType>
        </Output>
      </Outputs>
      <Indice>1</Indice>
      <Visibilite>true</Visibilite>
    </Service>
    <Service>
      <ServiceNom>Multiplication</ServiceNom>
      <Inputs>
        <Input>
          <InputNom>a</InputNom>
          <InputType>int</InputType>
        </Input>
        <Input>
          <InputNom>b</InputNom>
          <InputType>int</InputType>
        </Input>
      </Inputs>
      <Outputs>
        <Output>
          <OutputNom>RéponseMultiplication</OutputNom>
          <OutputType>int</OutputType>
        </Output>
      </Outputs>
      <Indice>2</Indice>
      <Visibilite>false</Visibilite>
    </Service>
    <Service>
      <ServiceNom>Soustraction</ServiceNom>
      <Inputs>
        <Input>
          <InputNom>a</InputNom>
          <InputType>int</InputType>
          <Value>5</Value>
        </Input>
        <Input>
          <InputNom>b</InputNom>
          <InputType>int</InputType>
        </Input>
      </Inputs>
      <Outputs>
        <Output>
          <OutputNom>RéponseSoustraction</OutputNom>
          <OutputType>int</OutputType>
        </Output>
      </Outputs>
      <Indice>3</Indice>
      <Visibilite>true</Visibilite>
    </Service>
  </Services>
  <Relation>
    <ServiceConnecte>
      <Nom>Addition</Nom>
      <IndiceIn>1</IndiceIn>
      <Nom_Input>a</Nom_Input>
      <Service_Entrant>Multiplication</Service_Entrant>
      <IndiceOut>2</IndiceOut>
      <Nom_Output>RéponseMultiplication</Nom_Output>
    </ServiceConnecte>
    <ServiceConnecte>
      <Nom>Addition</Nom>
      <IndiceIn>1</IndiceIn>
      <Nom_Input>b</Nom_Input>
      <Service_Entrant>Soustraction</Service_Entrant>
      <IndiceOut>3</IndiceOut>
      <Nom_Output>RéponseSoustraction</Nom_Output>
    </ServiceConnecte>
  </Relation>
</Contrat>
```

Figure 3. Schema XSD of the service contract

Then we parse this XSD file and generate the code (see Figure 4) that allows creating the execution entity for the end-user.

```
AttributService i21= new AttributService();
i21.nom = "b";
i21.type = "int";
Soustraction3.Inputs.Add(i21);
AttributService O20= new AttributService();
O20.nom = "RéponseSoustraction";
O20.type = "int";
Soustraction3.OutPuts.Add(O20);
Soustraction3.ServiceTete=true;
_liste.Add(Addition1);
Multiplication2.Suivant = Addition1;
Addition1.getInput("a").connextion=Multiplication2.getOutput("RéponseMultiplicatio
n");
_liste.Add(Multiplication2);
Soustraction3.Suivant = Addition1;
Addition1.getInput("b").connextion=Soustraction3.getOutput("RéponseSoustraction");
_liste.Add(Soustraction3);
}
public List<Service> liste
{
get { return _liste; }
}

}
}
```

Figure 4. Part of Code to generate the execution entity

Finally in the GUI, we will get the result in “Figure 5”

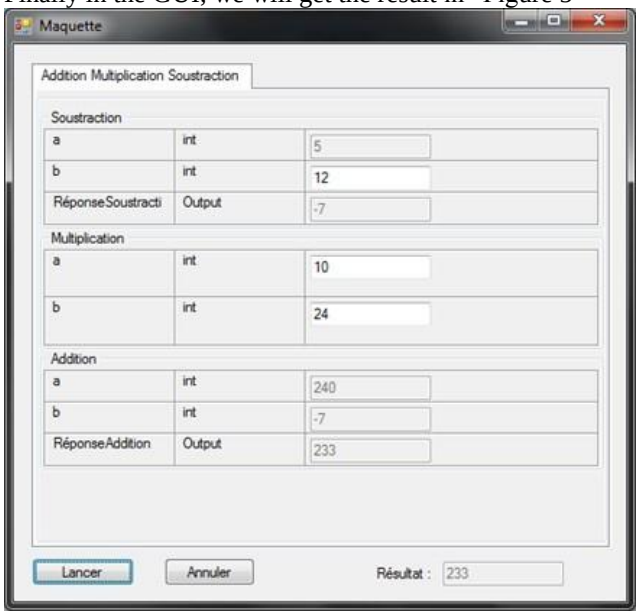

Figure 5. Result in the GUI

V. APPLICATION

We present below an application of our system through the following use case scenario.

This is a provider for customizing the web services specialized in real estate. For this, it has identified the WSs in its Knowledge Base; on the other hand, the following services can be claimed by a potential user:

- WS1: « Square Habitat » is a service of real estate
- WS2: « Mappy » is map service (Google/Apple)
- WS3: « BUS » : is a carpool service
- WS4: a service of local real estate agent

The available services are provided through the interface "Catalog" that the user can preselect and then create his own logic using these services through a GUI. The transaction is to find the corresponding service elements to achieve the composition and generate executable code available to the user.

"Figure 6" shows the interface that allow user to specify the desired services and their sequence by his own logic. For example, searching for properties, and then identifying the location of the real estate agent in order to find a way to get there by carpooling. So as to the context adaptation of the composition, the mediator must replace the concrete service entity by a "proxy", which is a meta-WS symbolizing the abstract service. The role of meta-WS is to identify and operate the concrete entities which are best suited at run time depending on their context.

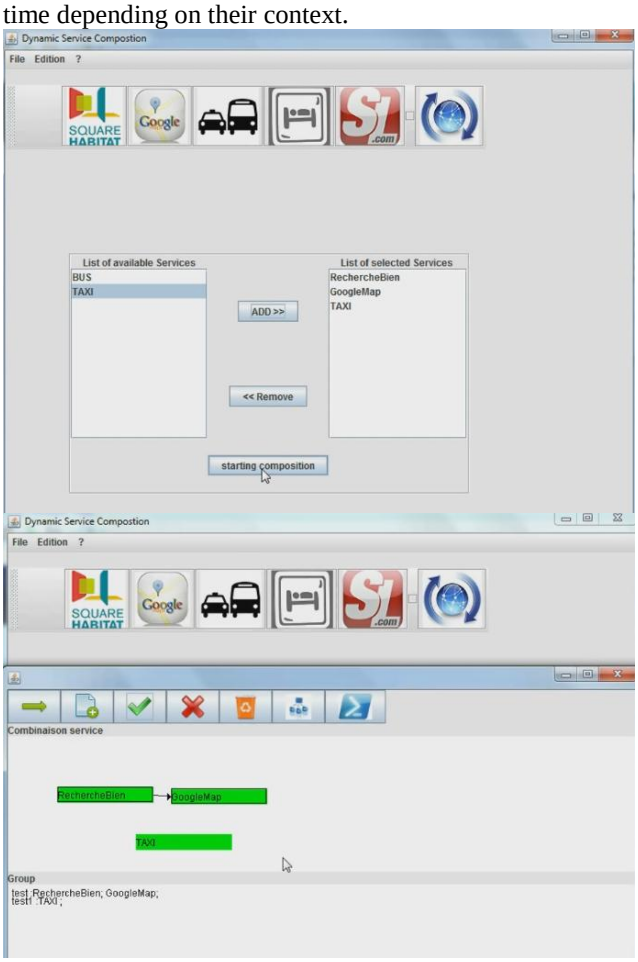

Figure 6. Application

VI. CONCLUTION AND FUTURE WORKS

In this paper, we propose a mediator-based architecture for dynamic service composition. We have clarified the main functionalities. Meanwhile, we show the potential capacity of this architecture to integrate the semantic web and to ensure the context adaptation. We presented a model (based on .NET/WCF) that we developed to validate our proposal and to provide a functional service composition tool ultimately. Furthermore, we should pay attention to the RESTful Web services which are lightweight. This

 ISBN: 978-1-61208-232-5

supposition, however, is going to be considered in our future work.

REFERENCES

[1] Web Service Business Process Execution Language Version 2.0. OASIS Standard, 11 April 2007. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf [retrieved: November, 2012]

[2] Web Services Choreography Description Language Version 1.0 , W3C Candidate Recommendation 9 November 2005, http://www.w3.org/TR/ws-cdl-10/ [retrieved: November, 2012]

[3] C. Peltz, Web services orchestration and choreography. IEEE Computer .36(10): pp.46-52 (2003)

[4] INRIA future Project Team Gemo. Management of Data an Knowledge Distributed Over the Web. http://www.inria.fr/rapportsactivite/RA2005/gemo/gemo.pdf [retrieved: November, 2012]

[5] OWL-S:Semantic Markup for Web Services.W3C Member Submission 22 November 2004.http://www.w3.org/Submission/OWL-S/. [retrieved: November, 2012]

[6] Kopecký, J., Vitvar, T., Bournez, C., and Farrell, J. (2007). SAWSDL: Semantic Annotations for WSDL and XML Schema. IEEE Internet Computing, 11(6), pp.60-67. doi: 10.1109/MIC.2007.134.(2007)

[7] http://www.wsmo.org/TR/ [retrieved: November, 2012]

[8] Web Service Modeling Ontology(WSMO),W3C Member Submission 3 June 2005. http://www.w3.org/submission/WSMO/. [retrieved: November, 2012]

[9] http://lsdis.cs.uga.edu/projects/meteor-s/ [retrieved: November, 2012]

[10] P. Rajasekaran, J. Miller, K. Verma, and A. Sheth, Enhancing Web Services Description and Discovery to Facilitate Composition, International Workshop on Semantic Web Services and Web Process Composition, Proceedings of SWSWPC (2004)

[11] Ponnekanti S R and Fox A., SWORD: A developer Toolmit for Web Service Composition. Proceeding International WWW Conference (11). (2002)

[12] IBM Alphaworks, BPWS4J, http://www.alphaworks.ibm.com/tech/bpws4j. [retrieved: November, 2012]

[13] The Protégé ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu/. [retrieved: November, 2012]

[14] Sirin E. OWL-S API. http://www.mindswap.org/2004/owl-s/api/. [retrieved: November, 2012]

[15] Macaya,C., Falchuk, B., Dana Chee Lin, F.J. Das, S. Ito, M. Komorita, S. Chiba, and T. Yokota, H.: Services Composition based on Next-Geeration Service Overlay Networks Architecture. In:New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference, pp.1-6. IEEE Computer Society,Paris (2011)

[16] Sivasubramanian, S.P., Ilavarasan, E., and Vadivelou, G.: Dynamic Web Service Composition: Challenges and Techniques. In: Intelligent Agent & Multi-Agent Systems, 2009. IAMA 2009.International Conference, pp.1-8. Philharmonic Luxembourg (2009)

[17] Radiant SAWSDL/WSDL-S Annotation Tool http://lsdis.cs.uga.edu/projects/meteors/downloads/index.php?page=1 [retrieved: November, 2012]

[18] Gomadam, K. et al., Radiant: A tool for semantic annotation of Web Services. in The Proceedings of the 4th International Semantic Web Conference , ISWC (2005).

[19] K.Gomadam et al., A faceted Classification-Based Approach to Search and Rank Web APIs, to appear in Proc. IEEE Int'l Conf. Web Services , IEEE Press (2008)

[20] Bachtarzi, F. Chemaa, S, and Chaoui, A. A G-Net based approach for Web service composition, Innovation in information & Communication Technology(ISIICT), pp.19-24 (2011)

[21] Xumin Liu and Hua Liu, Automatic Abstract Service Generation From Web Service Communities, Web Services(ICWS), 2012 IEEE 19th International Conference , pp.154-161 (2012)

[22] Yang Syu, Shang-pin Ma, Jong-Yih Kuo, and Yong-Yi FanJiang, Asurvey on Automated Service Composition Methods and Related Techniques, 2011 IEEE Ninth Internationl Conference on Services Computing(SCC), pp.290-297 (2012)

[23] Zhenpeng Liu, Junbao Liu, Jimin Li, Aiguo An, and Jianmin Xu, A model for Web Services Composition Based on QoS and Providers' Benefit, Wireless Communications , Networking and mobile Computing, 2009, pp.1-4 (2009)

[24] Lizhn Cui, Jian Li, and Yongqing Zheng, A Dynamic Web Service Composition Method Based on Viterbi Algorithm, ICWS, 2012 IEEE 19th International Conference, pp.267-271 (2012)

 ISBN: 978-1-61208-232-5