Developing Service Oriented Computing Model Based On Context-Aware

Hamid Mcheick*
University of Quebec At Chicoutimi, Computer Science Department
555 Boul De l'Universite, Chicoutimi (QC), Canada, G7H 2B1
*Corresponding author

Mohamed Dbouk
Department of Computer Science,
Faculty of Sciences (I), Lebanese University,
Rafic-Hariri Campus, Hadath-Beirut, Lebanon

Ahmad Karawash
Ecole Doctorale des Sciences et de Technologie, Lebanese University,
Rafic-Hariri Campus, Hadath-Beirut, Lebanon

Abstract: SOA and Cloud Computing are making major changes in the way companies build and deploy applications. The challenge is to meet the business expectation of faster delivery of new functionality, while at the same time maintaining control of application performance and availability across a growing network of service providers. SOA facilitates the development cycle by providing common features to everyone. However, SOA has some disadvantages such as the lack of information of what a service can provide and how can we discover it. When working with web services, the number of exposed methods or functions becomes a problem for developers. We do not need to deal with whole services if a developer needs to call one function. This article suggests and validates a new selected service model for the SOA. The layout presentation and the communication is described between client and services.

Keywords: Service Oriented Computing; Meta-Model; Web services; Cloud Computing.

Reference to this paper should be made as follows: Mcheick, H. and Dbouk, M. (2012) 'Developing Service Oriented Computing Model Based On Context-Aware', *Int. J. Communication Networks and Distributed Systems, Vol. X, No. Y, pp.000-000*.

Biographical notes: Hamid Mcheick is currently an associate professor in Computer science department at the University of Quebec At Chicoutimi (UQAC), Canada. He holds a master degree and PhD. in software engineering from Montreal University, Canada. Professor Mcheick is interested in software development and architecture for enterprise applications as well as in

separation of concerns (aspect, component, services, etc.). His research is supported by many research grants he has received from the Canadian government, University of Montreal, CRIM (Centre de Recherche informatique de Montreal), University of UQAM, and University of UQAC.

Mohamed DBOUK is currently a full time Professor, at the Lebanese university - Faculty of Sciences (I) - Department of Computer Science. He received a Bachelor's Honor" in Applied Mathematics (Computer Science) from the Lebanese University, Faculty of Sciences, and a PhD from Paris-Sud 11 University (Orsay-France), 1997. His was (2005-2007) the director of the "Faculty of Sciences, Section I", Hadath-Beirut. He is the Founder and Coordinator of the research master "M2R-SI: Information System" at the Lebanese University (2009). His research interests include Software engineering, Information systems, GIS, Cooperative and Multi-Agent Systems, Groupware. He participates in many international projects: Euro-Mediterranean/UNESCO Avicenna e-learning Project, UNDP / RBAS Project; Enhancement of Quality Assurance and Institutional Planning at Arab Universities (2002-2004).

Ahmad Karawash is currently a PhD student at the Lebanese University and at the University of Quebec At Chicoutimi (UQAC). He received a Bachelor degree in Applied Mathematics (Computer Science) from the Lebanese University, Faculty of Sciences, and master degree from Lebanese University, Ecole doctorale. His research interests include Web services, Distributed systems, GIS, Social network services, network and Mobile technology.

1 Introduction

In the last decade, there has been lot of changes in the way software are developed and deployed. We started by the Assembly language (early 1950), to C language to support the concept of modules (1972), in which a developer could split his application into different C files or modules, to object-oriented methodology and languages (1985: C++, 1996: Java). SOA began its first steps in 2000, to replace "Objects" by "Services". These services can be consumed like any function in a class. The SOA has overcome many challenges, mainly the interoperability and the reusability.

The software development practices have evolved a lot. Applications have become more distributed in terms of their physical execution and the development of components. Service-oriented approach has become an important alternative to traditional software development [Cloud, 2012]. Applications such as Yahoo and Google can be considered as success stories in the SOA implementation [Kiciman, and Livshits, 2010]. But also, SOA has some drawbacks, services are published over the Internet and ready to be called by clients' applications but their metadata are limited. A developer can pass many hours while trying to discover the functions provided by a service or by reading a document describing these functions. There is no automatic method to filter or search a service. This article proposes a method to publish the layout of each service and the layout of the desired service that a client application needs. We suggest a model for the communication between these two entities (layout of a service and the layout of a client application).

For example, a web service can be published over the internet, it can provide the prices of the stocks and the index of the local market, the regional markets, the European

market, and other functions can be provided to retrieve historical data about each market. Also we can have functions that display charts on how the prices are moving during the last month. Such functions are bounded and published in one service but a developer may need the price of one index in his country, so why does he need to worry about all these functions? Our model tries to solve this problem.

Such problem is frequent in real world, since large corporations tend to publish web services with lot of functions in one package or service. The reason is that these services may belong to the same application domain. We aim to build a logic layer that will "hide" the unused functions of one or more services and "show" the functions requested by a client application. As stated in this section, the functions needed by a client application are highlighted in its layout.

We propose two methods of implementing this model:

1. Materialized. The logic layer built between the service and the client application will be static; it is not updated at each service call.
2. Virtual or Cloud Computing. The logic layer built between the service and the client application will be dynamic; it is updated at each service call.

We explain the SOA concepts and reflection methodology related to our work in section 2. Section 3 describes related works in the service computing adaptation and composition. Section 4 describes our model of ASC. Our experimental work is detailed in section 5 in addition to a comparison between our model and previous models. Finally we present the conclusion and the future works in section 6.

2 Background

This section explains briefly the service-oriented architecture, web services, services types and refection that are used in our research.

2.1 Service-Oriented Architecture

Service-oriented architecture (SOA) is a way of developing distributed system to manage and organize communications between web services. SOA defines how parts of job are performed through interactions of several entities (programs). Description language is used to define interactions between services. Interactions are independent of each other.

SOA is a method of architecting an application as a set of cooperating services that all users want. The user can be a human user or a client application [Erl, 2012]. For example, when we need to buy an item using an online electronics shop, many services can be executed such as create order, check inventory, place order and track delivery. These services are controlled using XML [SOA Definition, 2012].

2.2 Web services

"A Web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided" [W3C, 2004].
Web service has three parts: SOAP, WSDL and UDDI, which are summarized briefly in this section.

SOAP - This protocol is mainly used to exchange information over HTTP and over the internet. It describes how a message can be crammed into an XML document, it illustrates how a SOAP message should be transported through the Web and it puts set of laws and conventions that must be followed when processing a SOAP message.

The SOAP message body is designed to carry textual information. This is referred to as payload [Panda, 2005].

The exchange of information can be in a synchronous, where exchange of information takes place in a request/response form, and asynchronous mode, where exchange of information between several applications uses the message queuing route.

WSDL - The Web Services Description Language (WSDL) is an extension of the Extensible Markup Language (XML); it provides a mixture of tags including a complete description of a service.

WSDL forms one of the core building blocks of web services [UDDI, 2011]. Web services involve 3 participants: service provider, service broker, and service requester. The requester can also be called the web service client. A provider can be a system providing services. A requester can be a system in need of this service. The broker is a system that helps both provider and requester to discover each other [WebService, 2012].

UDDI - Universal Description, Discovery and Integration (UDDI) are a specification for the XML-based registries to list and find services on the World Wide Web [UDDI, 2011]. Registries are the electronic databases that enable businesses to store and access the services in an XML format. A registry can be a public registry or a private registry. UDDI's goal is to promote online collaboration among the business in the world. UDDI is an XML document composed of businessEntity that describe the organization that provides the service, businessService which contains list of web services presented by the business entity, bindingTemplate that describes the technical side of the service and tModel (technical model) that store additional information about the service.

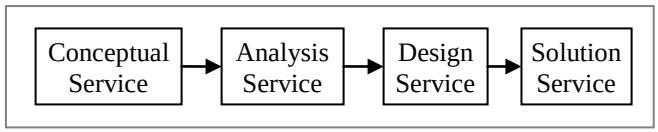
There are several steps in the entire Web Service process:

- Creation of the service by the service provider, and generation of WSDL document.
- The provider publishes the services in UDDI registries. The service publication includes the details of the provider.
- Publication of the location of the WSDL document in the registry exposes the web services to the community.
- The client side searches in the service registry by an SOAP-RPC invocation.
- To understand the web service semantics, the client side downloads the WSDL document and examines it.
- Service is invocated via a SOAP request over a transport protocol such as http.

2.3 Services evolution and types

Figure 1 illustrates the four important service-oriented life cycle transformation states. At the beginning, a service appears as an idea. Then, it becomes an analysis task. When the analysis phase is completed, a service evolves into a design entity. Finally, the service-oriented development life cycle produces a physical service ready to be deployed in production environment [Bell, 2008].

Figure 1 Services Evolution Life cycle.



2.4 Reflection

Reflection is the technology by which a program can view and modify its own structure and behavior. This is exactly how Reflection in Java and C# works. The ability to examine and change information about an application during runtime, offers huge potential [CodeSource, 2005]. In traditional software methodologies, developers are able to read/update their classes or components during design time, which means before running a program. With reflection, this manipulation is also available during run time, this leads to great flexibility to developers. Reflection is both a general term, as well as the actual name of the reflection capabilities in C# [CodeSource, 2005].

In many applications, we might need to save the users settings. When we get several settings, we can create a Settings class, which will handle loading and saving of the desired settings. Each time we need a new setting in our Settings class, we will have to update the Load() and Save() methods, to include this new setting. With Reflection, we can let the Settings class discovers its own properties and then loads and saves them automatically [Liberty, 2001].

3 Related works

Initially web services discovery was primarily syntactic. After development of semantic Web technologies, the proposed techniques for web service discovery became essentially semantic (level of semantic similarity between terms query and semantic web services description). This section describes three approaches of discovery services: Syntactic, Semantic and context-awareness.

3.1 Syntactic approach

The general principle of syntactic approach is to compare between query syntax based on user's keywords and syntactic Web Services description (WSDL).

In the UDDI approach [Newcomer, 2004], user or research program sends a query consists of keywords that is compared with registry keywords. The search result is a set of web services descriptions; the user selects the web service that best meets its requirements. This method returns a large number of results or conversely few results.

AASDU (Agent Approach for Service Discovery and Utilization) is another syntactic approach for discovering web services was proposed in [Palathingal and Chandra, 2004]. It is a multi-agent approach containing four components: a graphical user interface (GUI), an agent query analyzer (QAA), a system used to reference agents according to their expertise, where each agent has only knowledge of services related to their field of expertise, and the last component is the service module offering to providers.

Indeed to answer a query Q, the user enters his search query as string through GUI interface, this request is sent to agent QAA that extracts relevant keywords then selects a

set of expert agents which are linked to the agent composition. These agents invoke a service according to user's choice.

3.2 Semantic approach

Recent work has focused on semantic description web services, ontologies are used to model the semantic service representation, and it helps to institute semantic relations between concepts in a domain.

We mention the OWL-S approach [Matin et al., 2004] that uses the OWL-S ontology, the latter extends UDDI with semantic description of Web services.

In this method, discovery is based on Matchmaking algorithm, which allows finding web services descriptions that have semantic correspondence between functional parameters defined in descriptions services and those introduced in search query. Web services are then classified by semantic correspondence level between their output parameters and those cited in the query. If two services have the same correspondence level with the request in relation to their output parameters, a comparison on semantic correspondence level relative to the input parameters is then performed.

3.3 Context-aware Semantic Service Discovery

Context parameter was firstly used in the web services discovery process in 2006 by [Suraci1 et al., 2006].

The adaptation process is implemented in the mechanism of search services. This mechanism is based on the reference architecture based systems of services; the provider publishes its services on a server that the user sends a service request. We summarize the steps of this work in the following steps:

- The provider must publish in the context manager the context in which the service can meet and conditions.
- The provider must publish in the register the two descriptions of the basic and semantic service conditions of use service, and the reference of the context service.

The user must save its context (User Context) in the context manager (Context Manager) before he can make a request. This request may be a basic query expressed in low-level research language or query semantic expressed as a semantic language query of high level.

Then the user must send to the server requests and the pointer to its context. This information is stored in a server module named User. Once the user request is received, the search server service enables the filtering engine service based on three phases:

The basic filter: allows selecting the category of the service to be selected and so decreases the number of services which are going to be subjected to the second stage of the filtering.

The semantic filter: The result of this filter is a list of services which answer exactly the characteristics wished by the user, without the information of the context.

The filter of context: this last stage is decomposed into three phases. First, the server of research services (SDS) compares the service context and the waits of the user in terms of context of service. It compares the context of the user with the

conditions of use of the service, and then it compares the context of the environment with conditions of use.

4 Adapted Service Computing Model

4.1 Current situation

When a list of services is published on a web server in SOA architecture, any client application can access one or more of these services. To use one method or function of a service, a client application should reference the service that the method belongs to. Therefore all methods or functions of this service will be available to be used. This leads to problems since the interest is limited to only one method. Why should the developers of a client application get lost due to the huge number of methods available in a service when they only need one specific method or functionality?

Figure 2 shows the SOA model: Service providers publish their web services on a web server. Any client application can connect to this server and call one or more of these services.

In order to use method i of the service j, a developer might need to discover lot of unnecessary functions.

Figure 2 SOA Model

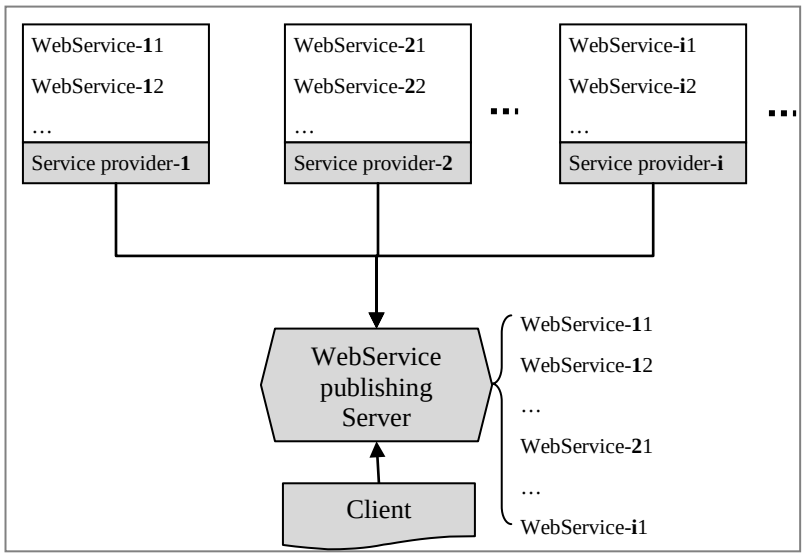
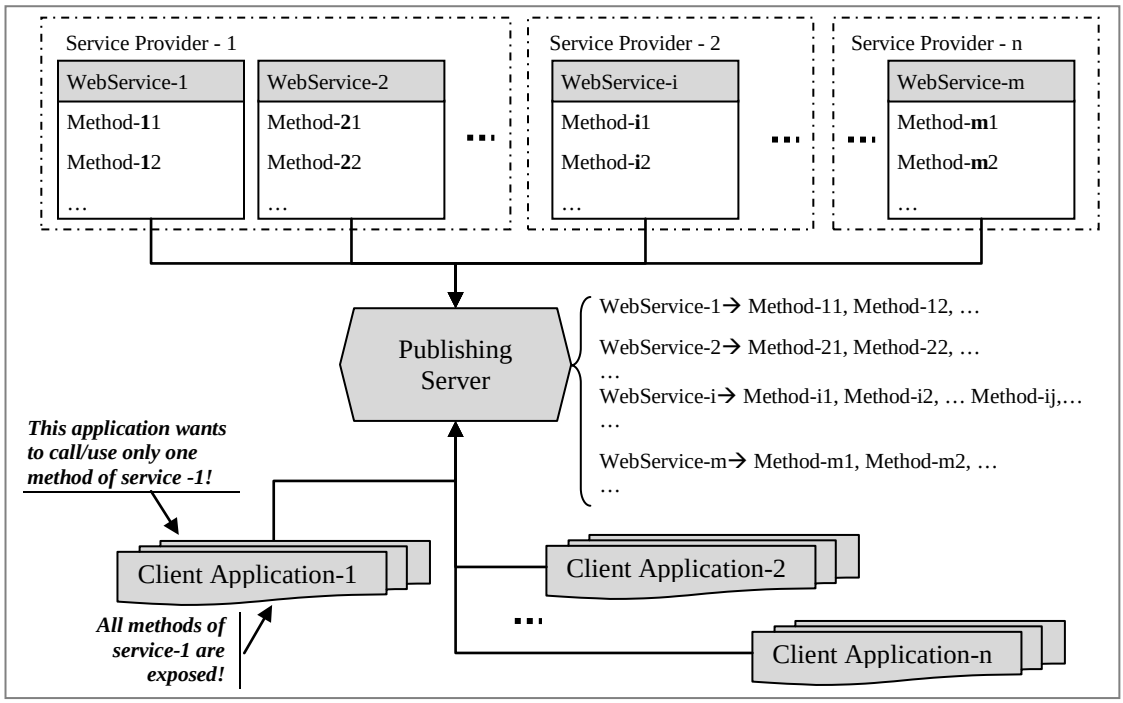


Figure 3 displays the current problem with the SOA architecture. In this figure, a developer is working on a client application “Application 1”. This application needs to call method i of the service 1. In this case, the developer will have all methods of service 1 available. This service might have hundreds of functions depending on its complexity or its business domain, while the developer needs only one method.

Another example illustrated in this figure, a developer is working on a client application “Application 2”. This application needs to call method i of the service 2 and method j of the service n. In this case, the developer will have all methods of service 2

and service n available. Similar to the first case above, these services might have hundreds of functions depending on their complexity or their business domain, while the developer needs only one method of each service.

Figure 3 Problems with the SOA Architecture.



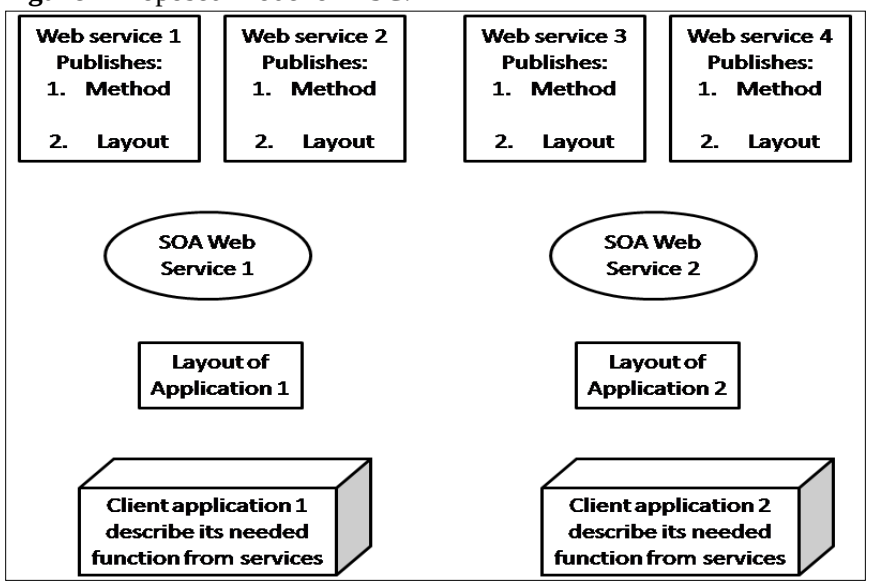
4.2 Our Adapted Service Computing Model

We are proposing a model to decompose a service and to adapt it to clients' needs. We suggest the following model to solve this problem. Our model aims to adapt the service(s) based on a client application and not in a unique way for all clients' applications. The steps of this model can be described as follows:

1. Each service should be published with its layout. For example: the service S1 is published with its layout Ct1.
2. The layout Ct1 should contain the following fields: Business Domain; Category; URL Function Name; List of the input parameters' types; Output type of the function; Dependency; Data constraint, example; the account number maximum length is 15 digits. The date format is mm/dd/yyyy; Username, some services may need credentials to be accessed; Password, some services may need credentials to be accessed. Description, example: this function returns the list of indices available in Beirut Stock exchange. Last update date; this is an indicator to the developer if the function is being updated frequently.
3. When an application needs to call a service, it should also provide its layout.

4. The layout of a client application should contain some (or all) of the following fields:
    a) Business Domain.
    b) Category.
    c) Function Name.
    d) Input parameters' types (mapping the service and the client application).
    e) Returned type.
    f) Web server URL, which represents the server where this service is hosted.
    g) Username, some services may need credentials to be accessed.
    h) Password, some services may need credentials to be accessed.
    i) The description.
    j) The last update date.

Figure 4 Proposed Model of ASC.



The layout of the client application may contain some of the above fields, as much as this layout is detailed, which means many properties are filled, then the matching with the layout of the service will be easier and accurate.

1. The client application layout explores the layout of the service.
2. A contract will be set between the service and the application. By contract, we mean an agreement of communication between both parties.
3. The layout of the service is a XML document.
4. The layout of the client application is a XML document.
5. An object model will result from the communication between the 2 layout files; this model is detailed in the experimental section.
6. The client layout is prepared manually by the application developer because he knows what he needs from a specific service published on the web.
7. The service layout is prepared manually by the service provider.

Using this object model, the client application calls the needed functions only and disregards all other methods (table 1). The layout of the service is of the form: Name→ Value

Example: a service contains 3 web methods:

1. Make Transfer.
2. Approve Transfer.
3. Add Beneficiary.

Table 1 Layout of service (description)

Name	Value
Domain	**Financial**
Category	**Banking**
URL	**http://192.168.1.111/MyService.asmx**
FunctionName	MakeTransfer
InputParam1	Integer
InputParam2	Integer
InputParam3	Decimal
OutputParam	Integer
Username	NA
Password	NA
Description	This function is used to transfer money between 2 accounts
LastUpdateDate	01/05/2011
Domain	**Financial**
Category	**Banking**
URL	**http://192.168.1.111/MyService.asmx**
FunctionName	ApproveTransfer
InputParam1	Integer
OutputParam	Integer
Username	user123
Password	12345678
Description	This function is used to approve a transfer operation
LastUpdateDate	01/05/2011
Domain	**Financial**
Category	**Banking**
URL	**http://192.168.1.111/MyService.asmx**
FunctionName	AddBeneficiary
InputParam1	String
InputParam2	String
InputParam3	Integer
InputParam4	String
OutputParam	Integer
Username	NA
Password	NA
Description	This function is used to add a beneficiary
LastUpdateDate	01/05/2011

If we have a client application that needs to calls the "MakeTransfer" and the "AddBenficiary" functions of this service. The model that will represent this request might be as shown in table 2.

Table 2 Request of clients

Name	Value
Domain	**Financial**
Category	**Banking**
URL	**http://192.168.1.111/MyService.asmx**
FunctionName	
InputParam1	Integer
InputParam2	Integer
InputParam3	Decimal
OutputParam	Integer
Username	NA
Password	NA
Domain	**Financial**
Category	**Banking**
URL	**http://192.168.1.111/MyService.asmx**
FunctionName	
InputParam1	String
InputParam2	String
InputParam3	Integer
InputParam4	String
OutputParam	Integer
Username	NA
Password	NA

.

4.3 Modifying WSDL file to implement our model: an alternative way

Another method for implementing our model is also available; we can use a modified version of the WSDL file generated for every web service. Mainly the WSDL contains the input, output and the function name of every function. We can add the additional fields such as the category and the domain to the WSDL file.

For sure, this will lead to a major change in how the WSDL is generated, that's why we focused on the first option. A second disadvantage of this method is that the services' providers should implement major updates to the existing web services.

5 Experimented works

The service adaptation model that we propose in this research can be implemented in 2 methods:

- Adapting services by searching the metadata using the Web Services Inspection Language.
- Adapting services by searching the metadata as string comparison.

5.1 Web Services Inspection Language

The Web Services Inspection Language (WSIL) is a service discovery mechanism [Modi, 2002]. WSIL allows us to go directly to the service provider and ask for the services it provides.

The WSIL specification does not define a service description language. WSIL documents provide a method for aggregating different types of service descriptions. Within a WSIL document, a single service can have more than one reference to a service description. For example, a single Web service might be referenced twice in a WSIL document: once directly via its WSDL, and again via its business service entry in a UDDI registry. References to these two service descriptions should be put into a WSIL document.

If multiple references are available, it is good to put all of them in the WSIL document so that the application that uses the document can select the type of service description that is compatible with and preferred by that application [Modi, 2002].

The WSIL specification serves two important functions:

1. WSIL defines an XML format for listing references to existing service descriptions. These service descriptions can be defined in any format, such as WSDL, UDDI, or plain HTML. The ability to link a WSIL document to one or more different WSIL documents allows us to manage service description references by grouping them into different documents and to build a hierarchy of WSIL documents. For example, separate WSIL documents can be created for different categories of services, and one primary WSIL document can link all of them together.
2. WSIL defines a set of conventions so that it is easy to locate other WSIL documents. The WSIL specification does not limit the type of service descriptions that can be referenced.

Two conventions make the location and retrieval of WSIL documents easy:

- Fixed-name WSIL documents. The fixed name for WSIL documents is inspection.wsil. The inspection.wsil file is placed at common entry points for a Web site. For example, if the common entry point is http://entrypoint.com then the location of the WSIL document would be http://entrypoint.com/inspection.wsil.
- Linked WSIL documents. References to WSIL documents can also appear within different content documents, such as HTML pages.

This work focused on string comparison between the service metadata and the client application metadata as discussed in the next section.

5.2 Text Search

During the demo, we will use the C# programming language. The demo will consist of:

1. A web service containing a number of web methods.
2. The layout of the service will be prepared as a XML file.
3. A windows application which represents the client application that needs to call the web service.

4. The layout of the windows application which represents in our model the needs of the application from the service or the specifications of the methods that the application needs to call.
5. A layer will be built (DLL); this layer contains the functions that were matched between the service methods and the requests of the application.
6. An object will be created in order to invoke the matched functions.

5.3 Comparison

As shown in the comparison table below, the main difference between our model and the models discussed in the related work section is that our model can be implemented during the analysis phase of the service development project or even after launching this service to the public, where as almost all previous models had to be implemented at design level.

Our model can be rapidly used by developers since it requires some XML knowledge only. Moreover, the most important advantage of our model is that the adaptation is per client. This means we do not have an adapted or composed model that all clients' applications should use, but in our model, the client defines his needed functions, and based on this definition, the service is adapted. Our model offers an "adaptation per client". The ASC model has one limitation which is the manual written of the service layout. If we have a service containing hundreds of functions, the preparation of this layout will take lot of time.

Table 3 Comparison between existing Model of compositiona and our model

Specification	Related Works	ASC Model
Implementation	Design Level	Design and Deployment Level
Difficulty	Depend on the Research	XML Basic Knowledge
Adaptation	One Model for All Clients	One Model Per Client

6 Conclusion and future works

The SOA technology is being adopted by the most software companies. The published services need an adaptation or a composition due to many reasons, mainly because when a service is published, limited information is published about what it can provide. When working with web services, the huge number of exposed methods or functions becomes a problem for developers. Also there is no automatic method to filter or adapt services based on client needs. As for the WSDL file that is automatically created with each web service, it is hard to read and extract information from it.

We propose a model that will adapt a service per client needs. The result of this model is a logic layer that shows the functions or services requested by a client and hides unwanted functions.

Finally, there are a number of unaddressed issues, which, once solved, may turn out to be very helpful. The automatic update of the DLL was not discussed. If the service's functions were modified due to a software updates, the output of our model which is a

DLL, should be dynamically updated too. We didn't focus on the Information Retrieval methods on metadata (part 1 of the Experimental Work section).

7 Acknowledgment

This work was sponsored by Natural Sciences and Engineering Research Council of Canada (NSERC), the University of Quebec at Chicoutimi (Quebec), Canada and the Lebanese university, Beirut, Lebanon.

8 References

[Cloud, 2012] Cloud Computing, www.gomez.com. deVadoss J. Understanding Service Composition, Part I: Dealing With Workflow Across Services. *SOA Magazine Issue XXXVIII*. http://www.soamag.com/I38/0410-2.php, April 8, 2010.

[Erl, 2012] Erl T. *What is SOA: an Introduction to Service Oriented Computing*, http://www.whatissoa.com, SOA System, 2012.

[CodeSource, 2005] http://www.codersource.net/microsoft-net/c-basics-tutorials/c-net-tutorial-reflection.aspx, Code Source, 2005.

[Kiciman, and Livshits, 2010] Kiciman E., and Livshits B. AjaxScope: A Platform for Remotely Monitoring the Client-Side Behavior of Web 2.0 Applications , Emre Kiciman and Benjamin Livshits, Microsoft Research Project, *ACM Transactions on The Web*, Vol. 4, No. 4, Article 13, Pub. date: September 2010.

[Liberty, 2001] Liberty J. *Programming C#,* Chapter 18, Attributes and Reflection, O'Reilly, 2001.

[Martin et al., 2004] Martin D., Burstein M., Hobbs J., Lassila O., McDermott D., Mcllraith S., Narayanan S., Paolucci M., Parsia B., Payne T., Sirin E., Srinivasan N., Sycara K. Owl-s : Semantic markup for web services. *Technical report*, W3C, 2004.

[w3c, 2004]: web service definition, http://www.w3.org/TR/wsa-reqs/, (2004).

[Modi, 2002] Modi T. WSIL: Do we need another web service specification? http://www.webservicesarchitect.com/content/articles/modi01.asp. Web Services Architect, 2002.

[Newcomer, 2004] Newcomer E. Understanding Web Services- XML, WSDL, SOAP and UDDI, chapter 5, *Finding Web Services : UDDI Registry*. *Addison Wesley Professional*, May, 2004.

[Palathingal and Chandra, 2004] Palathingal P., and Chandra S., Agent approach for service discovery and utilization. *In HICSS*, 2004.

[Panda, 2005] Panda D. An Introduction to Service-Oriented Architecture from a Java Developer Perspective. http://onjava.com/pub/a/onjava/2005/01/26/soa-intro.html, *O'Reilly* 2005.

[Suraci1 et al., 2006] Suraci1 V., Mignanti S., and Aiuto A., Context-aware Semantic Service Discovery. *University of Rome "Sapienza*", Department of computer and system sciences, 2006.

[UDDI, 2011] Universal Description, Discovery, and Integration (UDDI). http://www.service-architecture.com/web-services/articles/universal_description_discovery_and_integration_uddi.html, 2011 Barry & Associates.

[SOA Definition, 2012] Service-oriented architecture (SOA) definition, http://www.service-architecture.com/web-services/articles/service oriented_architecture_soa_definition.html, Barry & Associates, 2012.

[Bell, 2008] Bell M. Service Oriented Modeling, John Wiley & Sons, Feb 25, 2008 - 384 pages.

[WebService, 2012] Web Services explained. http://www.service-architecture.com/web-services/articles/web_services_explained.html, 2012 Barry & Associates.