International Journal of Information Sciences and Application.
ISSN 0974-2255 Volume 5, Number 1 (2013), pp. 1-10

http://www.irphouse.com

Web Services Stepping Ahead

Shweta Roy[1] and Dr Ranjit Singh[2]

[1]Department of Information Technology,
[2]Department of Electronics Communications Engineering,
Ajay Kumar Garg Engineering College, 27 Km Stone, NH-24,
P.O. Adhyatmic Nagar, Ghaziabad 201009 UP India
[1]sguddr@gmail.com, [2]ranjit.2000@gmail.com 9868 041 558

Abstract

This paper presents a short introduction to web services. The current emphasis on Service-Oriented Architectures has put the spotlight on web services, but it's easy to get lost in all the information being bandied about. Web services extend the World Wide Web infrastructure to provide the means for software to connect to other software applications. Applications access Web services via Web protocols and data formats without worrying about how each Web service is implemented. Web services combine the best aspects of component-based development.

Keywords: Service oriented architecture, Web Services, Web protocols, Web Service Description Language

INTRODUCTION

Information available on website is intended for humans to access directly but information available through web services are meant for software to access despite the fact that the humans may be using it. Even though web services rely heavily on existing web technologies they do not have direct relationship with HTML and web browsers.

A web service is a kind of software that is accessible on the Internet. It makes use of the XML messaging system and offers an easy to understand, interface for the end users.

There are lots of buzz about web services now-a- days and companies are using them for enterprise applications. To put it simply, it is another

distributed computing technology like CORBA and RMI. They allow us creation of client server applications [7].

For example let us suppose a database with up-to-date weather information is kept in the server located at UK. It can be made available to anyone in the world through web services. This information can be published through web services and given the ZIP code. It will thus provide the information for that ZIP code.

Client who wants to access weather information would contact the weather web service and send the service request. The server will give service response which will give the information of following Zip Code.

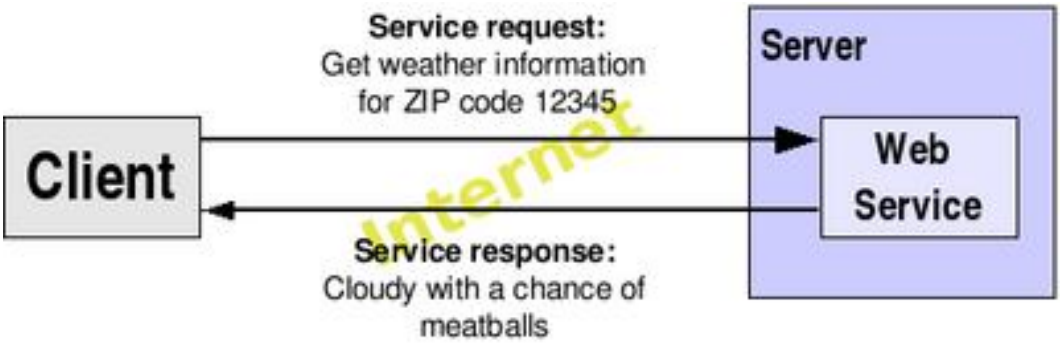


Figure 1

It can be done with other distributed technologies also like RMI, CORBA, EJBS etc but web services have some advantages over them.

- Web services are platform independent and language-independent. Both client programs may be coded in C++ and running on Windows while web services may be coded in java and running on Linux.
- Most web services are using HTTP for transmitting messages (service request and response messages). When we are making internet based applications, it becomes easy to manage through firewalls.
- But where there is boon there is some bane also. Platform independence and dependency on XML incurs overhead which is not present in binary coding. Web sevices are not so versatile also till now as other distributed technologies.
- Since, Web services are loosely coupled distributed system; this makes it most fit for internet applications.

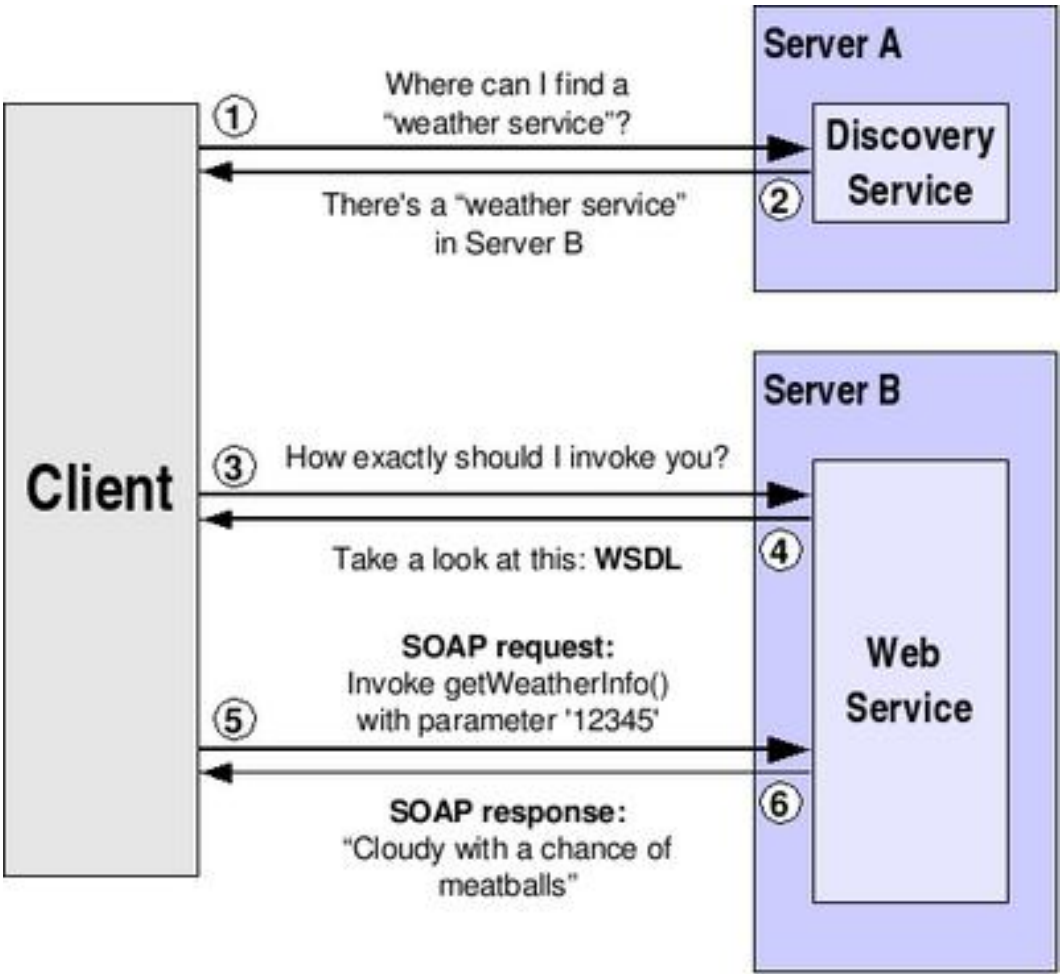
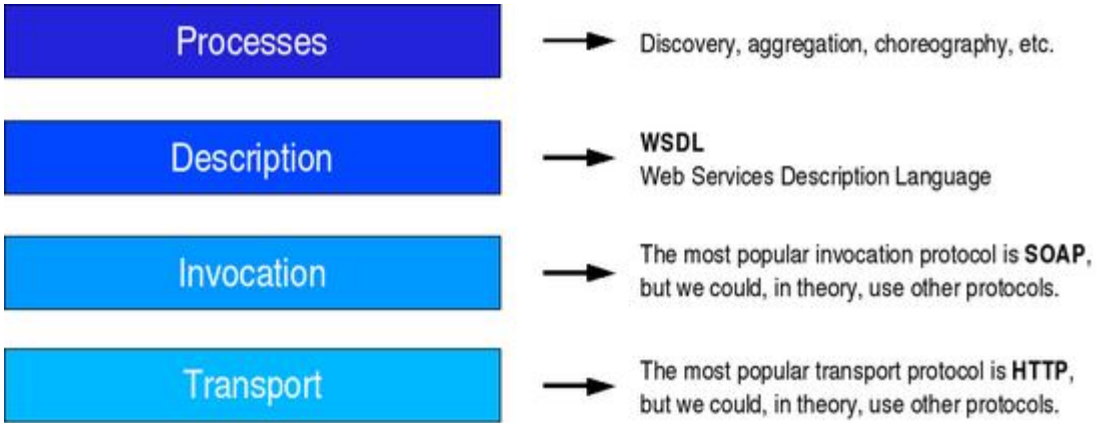


Figure 2

BASICS OF HOW TO INVOKE A WEB SERVICE.

- Clients do not have information about where to find the web service which will give the respective weather information. First step is to discover the service which meets our requirements which is done through sending request to discovery service which is again a web service (yellow pages).
- Discovery service will reply where to find the service which will meet our requirements.
- Now although we know the location but do not know how to invoke the web services which is done actually through WSDL (Web Service Description Language). Web service replies in WSDL language.
- Now when we know where web service is and how to invoke it, service request is sent through language called SOAP (simple object access protocol).
- Web service gives SOAP reply to corresponding SOAP response.

WEB SERVICE ARCHITECTURE



- Service process- This part of architecture involves lots of web services. Discovery process is also part of it through which we find web service according to our requirements.
- Service description: One of the most interesting features is that web services are self-describing. It is written in WSDL which says what the web service is for, and how to invoke it.
- Service invocation: It is done through SOAP request and response messages which is passed between client and server.
- Finally to transmit the message HTTP (hyper text transfer protocol) is used.

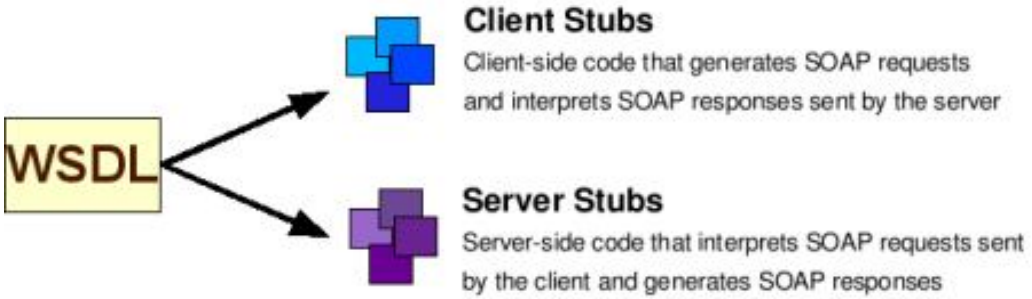


Figure 3

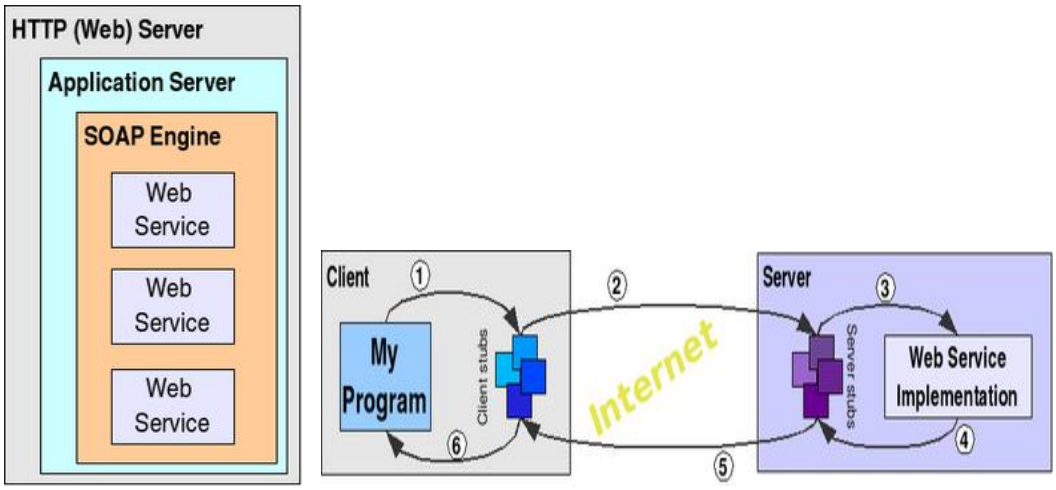


Figure 4

WEB SERVICES CHALLENGES

There are many challenges emerging with this new technology. The technology is itself in very nascent stage so during the process of evolution, many a milestone has been set.

Main challenge involved with web services is how to choose best web services available amongst so many web services performing similar functionality and how to automate the execution. A planning system can be thought of as the one performing user-defined goals [8].

Services are made up of operations which can be seen as actions that a planning system can perform on the world. However from the planning point of view, these actions are not fully specified. The problem of planning for web services is further complicated by the fact that the information describing the domain is distributed, heterogeneous and incomplete [5].

Heterogeneity is a problem both in terms of the data that services input and output and the description of the services themselves. There is no common high level data model between services. Two logically equivalent actions may be called differently in different services and require different patterns of interactions prior to their execution.

For example, one might need to perform a “login” action followed by “getCatalogue” action in one service before executing “purchaseItem”, while another service the logically equivalent action “submit purchase order” can be executed without executing any prior login procedure.

Our approach to planning of web services is pragmatic one based on information that is currently available in service interface definitions. An attempt to perform planning is based solely on information available in the service descriptions.

The set of operations provided by a service is available in WSDL description.

The i/o signature tells what type of document needs to be provided in order to execute it as well as types of document that will be returned upon execution.

It also gives description of the possible combination of the service that is order of execution.

But this gives only static description of the functionality of the service while the service process definitions provide procedural information to the planner.

Service classification information is found in UDDI service registries in which services are classified according to IT industry segment, provider and location. Meta data becomes useful to the planner when searching for relevant services and inferring similarity between them.

In order to fulfil the goal of automated planning of web services [4], the work has been divided into two parts that is work on type matching algorithm capable of discovering semantic equivalence between similar data type, and a service composition algorithm for use in composing service to achieve information goals.

TYPE MATCHING

The service is composed from its operation definitions. For this, we need to be able in some way to match different data-types such as goal with various service output. In this, we need to tackle the problem of data heterogeneity which is to decide if under some mapping the data described by one data type can be substituted for that described by another. Thus if we take output produced by one service, map it and use it as input to another service [3].

The data type has set of values that can be considered as representing different states of the world. For example when receiving message of type “person” with the name equal to “Mohan” and age equal to”21” we can interpret that there exists a person name Mohan and age 21. If another message of type “university student” with name and age as before, we can interpret it as saying that there exists a person Mohan whose age is 21 and he goes to university. The second message describes a smaller set of possible worlds than first.

Now if we need to fulfil a goal of type person then the instance of the message “university student” can be used to provide the required information. If however we require an input of type “university student” and have a message of type “person” the reverse is not possible as we do not know whether or not the instance to which the message refers is a university student or a school student and so on. Thus in order to be able to map from one data type to another we require that the latter describes the superset of possible output worlds that can be described by the former.

Now in our algorithm when we compare the goal type t (goal) to a particular service output type t(out) we require that t(goal)□M t(out), which is to say that all documents conforming to output type also conform to the goal

type after certain mapping M has been applied to them.

```
<Weather>
<Temperature type="decimal"/>
<Location type="string"/>
</Weather>
```

with a restriction that value of field “Location” should be “Adelaide”, should match against a schema such as:

```
<DailyWeather>
<LocalConditions>
<AmbientTemperature type="decimal"/>
<Rainfall type="decimal"/>
</LocalConditions>
<Address>
<City type="CityNames"/>
<State type="StateNames"/>
</Address>
</DailyWeather>
```

where:

```
<simpleType name="CityNames">
<restriction base="string">
<enumeration value="Adelaide"/>
<enumeration value="Melbourne"/>
<enumeration value="Perth"/>
....
</restriction>
</simpleType>
```

This is because the information required by the first can be found from within the second, i.e. the values for “AmbientTemperature” and “City” in the second can be mapped to “Temperature” and “Location” in the first. Note also the fact that the value “Adelaide” (which is a restriction on the field “Location” in the goal), is one of the possible values of the type “CityNames” in the output type, i.e. some instances of the output type adhere to the value restrictions in the goal.

SERVICE COMPOSITION ALGORITHM

Service composition algorithm is devised to compose and execute service operation to retrieve desired information. The algorithm takes as input the goal to be achieved and searches a UDDI directory for all services which are capable of outputting documents of sufficient similarity to the goal, using the type matching algorithm described previously [1,2].

The service interface with the most similar output is selected first. If there is more than one implementation of that interface, the algorithm will select one of them based on meta-data values. It then attempts to execute the particular service operation that produces the desired output. Before doing so, it must create the required input document. It starts by using the immediately available information, such as that given in the goal, the local information, and past input and output documents if they exist.

If the available information is not sufficient, the algorithm must again search the outputs of other services, i.e. the procedure calls itself recursively. Generally, not all of the data required to fill the input document will be contained in a single source, thus the process repeats on sub-elements of the input document until a complete document is produced or a search limit is exceeded. Having generated an input document, the algorithm attempts to invoke the operation.

If it does not produce the desired output, the algorithm rolls back certain decisions made when creating the input and tries again. The heuristic guiding this search can be based on the confidence the algorithm had in its decision at each point, i.e. the quality of the match. If after a “reasonable number” of attempts, the operation still can’t be executed, then the problem may be the data given as input to the previous (successfully completed) operation.

Thus the system either tries to re-execute the previous operation with different inputs, or gives up on the service altogether and searches for a new way of achieving the goal.

The search tree created by the above algorithm can be seen as an AND-OR tree, where the “OR” branches represent different ways of creating an input, and the “AND” branches represent combinations of service outputs that together produce an input. Leaves in the tree represent data found to be available locally. The execution algorithm described above performs a bounded best-first search through the tree, where the bound sets a limit on the number of failed execution attempts allowed for completing a given sub-tree. The execution bound is decremented for each level of decent in the tree.

This algorithm assumes that all of the operations within each service are atomic, and that the service to which they belong is stateless i.e. there are no ordering constraints on the executions of operations within a given service. In some cases, this assumption may be false, and the exact ordering of operations may be critical for the correct execution of services. For example a service might require that a “login” operation is performed prior to executing a “getStockQuote” operation.

The algorithm described above would never try to invoke the former operation and thus would never be able to successfully execute the latter. In some cases, such information may be available however in the form of service process descriptions. Such process descriptions may even provide additional information regarding the flow of data between operations within a service (i.e. fields in input and output documents that refer to the same value) [6].

CONCLUSION

Web services are in very initial stage of development. Though lot of services have been developed and are being used, no common platform and standard exists. W3C web service architecture working group is working towards this [6], [9].

Once there is common semantics and ontology for services developed by different vendors, implementation of type matching and service composition will become feasible.

REFERENCES

[1] Carman, M., and Serafini, L. 2003. Planning for web services the hard way. InWorkshop on Service Oriented Computing,International Symposium on Applications and the Internet (SAINT-2003). IEEE Computer Society Press.

[2] Carman, M.; Serafini, L.; and Traverso, P. 2003. Web service composition as planning. In Workshop on Planning for Web Services, 13th International Conference on Automated Planning and Scheduling (ICAPS 2003). Fellbaum, C., ed. 1998. WordNet: An Electronic Lexical Database. The MIT Press.

[3] McDermott, D. 2002. Estimated-regression planning for interactions with web services. In AI Planning Systems Conference.

[4] McIlraith, S., and Son, T. 2002. Adapting golog for composition of semantic web services. In Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002). Morgan Kaufmann.

[5] Mecella, M.; Pernici, B.; and Craca, P. 2001. Compatibility of e-services in a cooperative multi-platform environment.In 2nd VLDB Workshop on Technologies for e-Services (VLDB-TES 2001). Springer.

[6] Thakkar, S., Knoblock, C. A., Ambite, J. L.; and Shahabi, C. 2002. Dynamically composing web services from online sources. In Workshop on Intelligent Service Integration, The Eighteenth National Conference on Artificial Intelligences(AAAI).

[7] Shweta Roy, Evolution of Middleware Technology and Its Widespread Applications, AKGEC Journal of Technology, Vol.1, no.1, January 2010, pp 33-38 (ISSN 0975-9514).

[8] Shweta Roy & Dr Ranjit Singh, Web Services: Composition and Integration for Easy Use, , International Journal of Computer Science and Communication, Vol 2, no.1, March 2011, pp 259-264 (ISSN 0973-7391)

[9] Shweta Roy, "Neural Network Based Solution for Choice of Best Web Services" published by Lambert Academic Publication House, Germany , 2012. ART1 (adaptive resonance network) based neural network has been used

Shweta Roy obtained B.Sc Engineering in Computer Science and Engineering from Magadh University, Gaya in 2001.

Since last six years, she is teaching at Ajay Kumar Garg Engineering College where, she is an Assistant professor in the Department of Information Technology. Ms Roy submitted M.Tech Thesis to the Gautam Budh Technical University in the area of Middleware Web Services.

She has passion for teaching and has taught a number of courses namely Computer Networks, Compiler Design, Software Engineering, Automata Theory, Java Programming and C programming Concepts. Published two research papers.

Lambert Academic Publication House, Germany published the book, "Neural Network Based Solution for Choice of Best Webservices" in 2012 based upon her MTech thesis.

Dr Ranjit Singh obtained BTech., MTech and Ph.D degrees from the Indian Institute of Technology, Kanpur in 1969, 1971 and 1976 respectively. He specialized in the area of Electronic circuits and devices.

Published large number of technical papers in IETE journals in addition to in-depth technology-reviews covering emerging trends in Communications and information technology.

Since last four and half years, he is teaching at Ajay Kumar Garg Engineering College where, he is a Professor in the Department of Electronics & Communications Engineering.

He has abiding passion for teaching and research. Currently guiding MTech. and PhD scholars besides supervising BTech projects. He is Life Fellow of the IETE and attended international conferences held in France, Singapore, USA, Hong Kong and Nepal. Daily practices advance meditation.