Fall 2015

A Survey Paper on Service Oriented Architecture Approach and Modern Web Services

Kamala Manasa Dhara
Governors State University

Madhuri Dharmala
Governors State University

Chamma Krishan Sharma
Governors State University

ABSTRACT

Service-Oriented Architecture is an architectural design pattern based on distinct pieces of software providing application functionality as services to other applications via a protocol. It is a collection of micro-services which are self-contained and provides unit functionality. The architectural style has the following essential core features which are inter-operability, service abstraction, service discovery, service autonomy, service statelessness re-usability, loose coupling. Service-oriented architectures are not a new thing. The first service-oriented architecture for many people in the past was with the use DCOM (uses RPC – Remote Procedural Calls) and CORBA (uses IIOP protocol) but because of the lack of standards and also with the advent of modern web development (Web 2.0) and the use of mobile phones and their penetration service oriented architecture is being implemented as Web Services (uses mainly HTTP/HTTPS) protocol.

Most common implementations of Web Services can be done as SOAP (Simple Object Access Protocol)-based which essentially is a HTTP/HTTPS POST with an XML payload in it. SOAP based web services expose service interface using WSDL (Web Service Description Language) and there is a pre-defined contract via XSD (XML Schema Definition) between the service being exposed and the client side that consumes this service. The other most popular lightweight implementation of web services is using RESTful (Representational State Transfer) architecture where the payload is in JSON (Java Script Object Notation) / XML and uses RESTful style of communication to access resources on the server. So any application written in any language for example C# or C++ or C or Groovy or Java that can make a HTTP call should be able to access the services and since the data is in XML/JSON they can make a sense of data and this way we can re-use services and be inter-operable.

The goal of our survey is to delve deeper into SOA principles, key constituents and how Web Services - implementation of SOA has taken this into such a wide spread usage and created a phenomena and various technologies that can be used to develop/consume web services and also about the protocols being used and some common use cases in building re-usable and scalable application architectures using web services.

KEYWORDS: Service Oriented Architecture, SOAP, RESTful, DCOM, CORBA, WSDL, XML/JSON.

TABLE OF CONTENTS

1. Introduction…………………………………………………..…….1
2. Evolution of SOA…………………………………………..…….3
3. Service Orientation Principles……………………………..…..…4
4. Fundamental Building Blocks of SOA……………………..…….6
5. Top-Down View of SOA Constructs……………………..…...…7
6. SOA Myths and Facts……………………………………………8
7. Web Services……………………………………..…………9
8. Service Implementation…………………………………………9
9. SOAP vs. RESTful………………………………………...11
10. Web Service Interaction…………………………………...12
11. Steps to Create Web Service……………………………………12
12. Web Service Roles…………………………………………………13
13. Usage Examples…………………………………………...13
14. SOA Tools……………………………………………………14
15. Demo on Web Services created used RESTful and SOAP……......14
16. References…………………………………………………....15

1. INTRODUCTION

Service-Oriented Architecture is an architectural design pattern based on distinct pieces of software providing application functionality as services to other applications via a protocol.

A **Service-Oriented Architecture** is essentially a collection of services that communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity.

The **Service-Oriented Architecture** for Web Services has:

- a standard way for communication
- a uniform data representation and exchange mechanism
- a standard meta language to describe the services offered
- a mechanism to register and locate web services-based applications

Service Oriented Architecture (SOA) is a approach used to make a structural engineering architecture based upon the utilization of services. Services, (for example, RESTful Web Services) do some little capacity function, for example, creating information, producing data, approving a client, or giving basic expository services.

In addition to building and exposing services, SOA can influence these services again and again inside applications (known as composite applications). SOA ties these services to organization, or separately influences these services. Consequently, SOA is truly about settling existing architectures by tending to the vast majority of the significant frameworks as services, and abstracting those services into a solitary space where they are shaped into arrangements.

SOA is known to give both time-to-market preferences, and business nimbleness. The utilization of orchestration engines, or utilizing improvement situations that influence services and SOA, permit the individuals who manufacture applications to do as such rapidly, since the services give quite a bit of what the application requires. This gives the time-to-market advantage.

Basic in idea, SOA is likewise a best practice to alter broken architectures. With the wide utilization of models, for example, Web services, SOA is being advanced as the most ideal approach to convey design readiness to your venture, that is, whether you do SOA accurately. The issue has been that the ways that undertakings influence SOA as a compositional example changes enormously from big business to-big business. Consequently, the ROI from moving to SOA has gone from awesome triumphs, to altogether disappointments.

SOA is a substantial way to deal with tackle a large number of the structural issues that ventures confront today. Notwithstanding, the individuals who actualize SOA normally take a gander at it as something you purchase, not something you do. Subsequently, numerous SOA ventures are about obtaining some innovation that is sold as 'SOA-in-a-crate.' You get something-in-a-container, however not SOA, and that just adds to the issues.

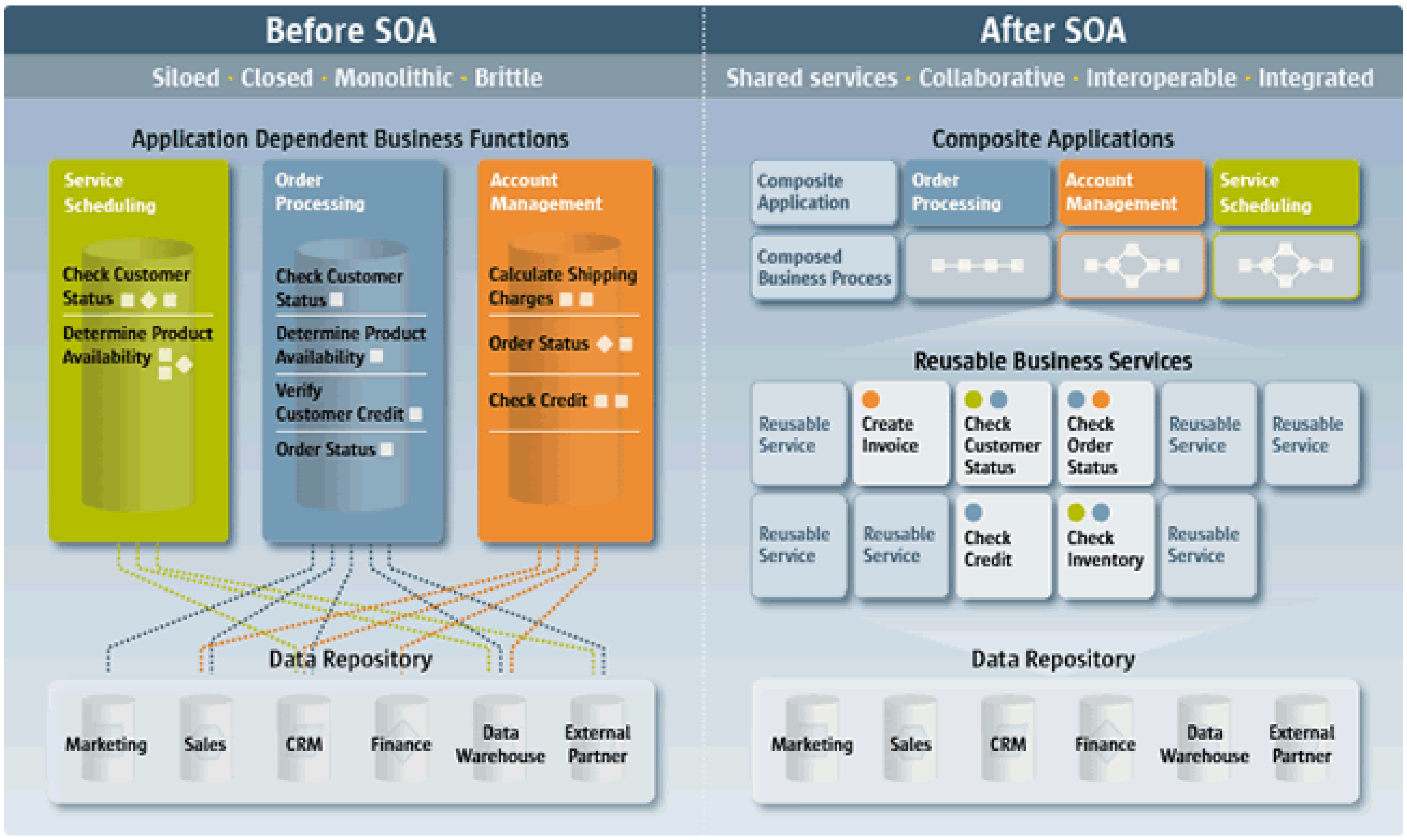
A SOA does not necessarily require the use of Web Services – Web Services are, for most organizations, the simplest approach for implementing a loosely coupled architecture. In the past, loosely coupled architectures have relied upon other technologies like CORBA and DCOM or document-based approaches like EDI for B2B integration. Many of these technologies are still in widespread use and are being augmented, replaced or extended with Web Services.

SOA is not particular to any innovation. Actually, if done accurately, SOA frameworks can without much of a stretch alter an excess of diverse advancements. Executing services situated structural planning uses basic conventions for different applications and services to cooperate. The following is a list of some of the things that play into service oriented architecture:

- XML Web Services (SOAP, basic Http, wsHttp)
- TCP/IP Services (binary)
- REST (XML and JSON)
- WCF (Windows Communication Foundation)

EXAMPLE: **Business Processes**

Service-oriented architecture (SOA) provides methods for systems development and integration where systems group functionality around business processes and package these as interoperable services. An SOA infrastructure allows different applications to exchange data with one another as they participate in business processes.



Source: http://www.logimethods.com/enterprise-soa-soe.php

2. EVOLUTION OF SOA

Service Orientation (SO) is the natural evolution of current development models. The, 80s saw object-oriented models; then came the component-based development model in the 90s; and now we have service orientation (SO). Service orientation retains the benefits of component-based development (self-description, encapsulation, dynamic discovery and loading), but there is a shift in paradigm from remotely invoking methods on objects, to one of passing messages between services. Schemas describe not only the structure of messages, but also behavioral contracts to define acceptable message exchange patterns and policies to define service semantics. This promotes interoperability, and thus provides adaptability benefits, as messages can be sent from one service to another without consideration of how the service handling those messages has been implemented.

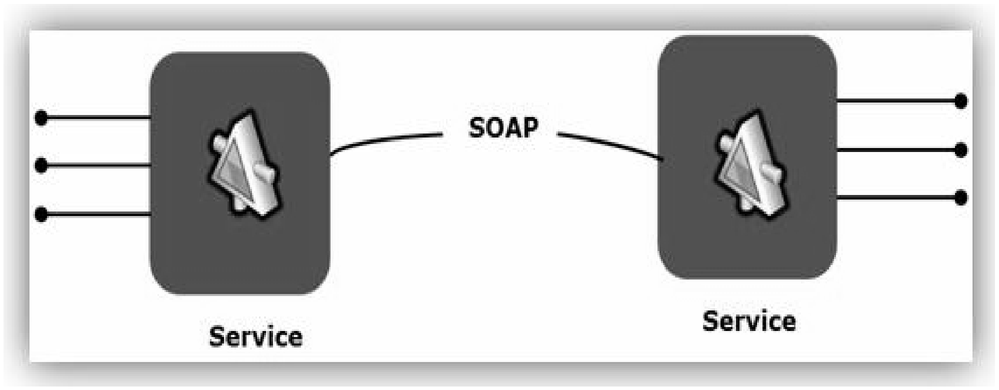
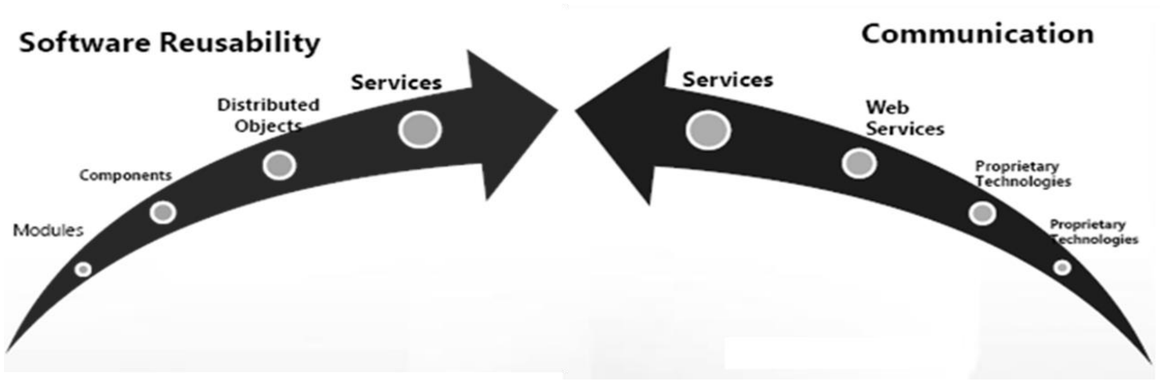


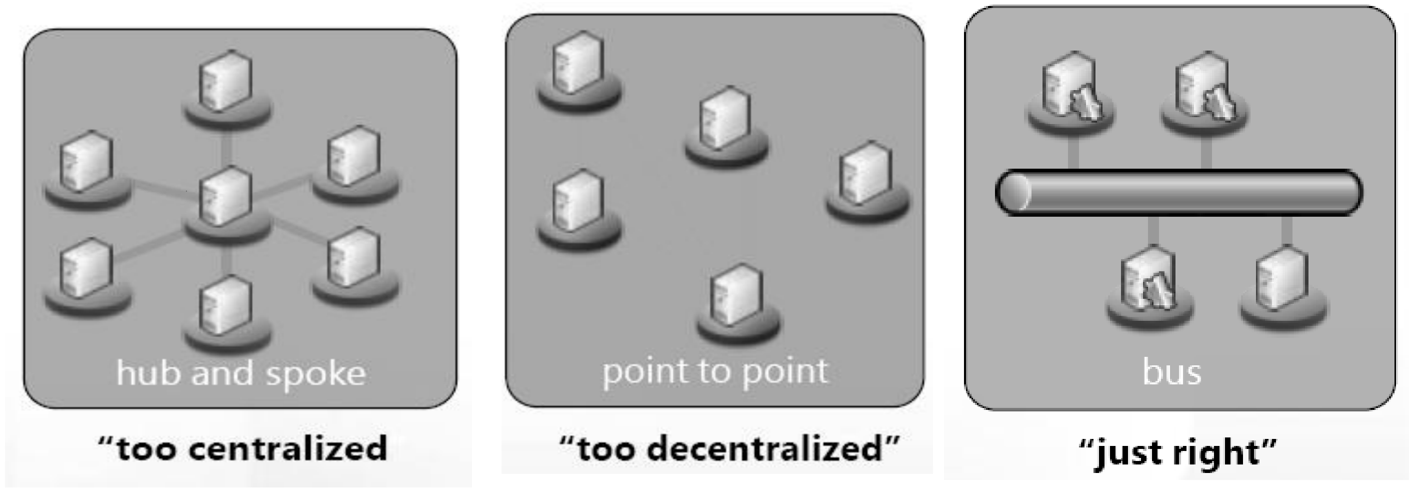
Figure. Simple SOAP-based communications between Web Services

Source: https://msdn.microsoft.com/en-us/library/bb833022.aspx

Service Orientation gives a transformative way to deal with building circulated programming that encourages inexactly coupled mix and flexibility to change. With the approach of the WS-* Web Services, structural planning has made administration situated programming advancement achievable by ethicalness of standard improvement apparatuses backing and wide industry interoperability. Albeit most regularly actualized utilizing industry standard Web Services, Service Orientation is autonomous of innovation and its engineering examples and can be utilized to join with legacy framework also.



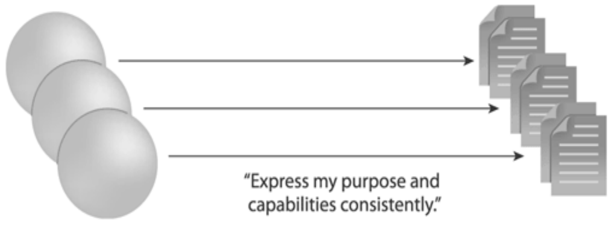
Source: Integraph Corporation



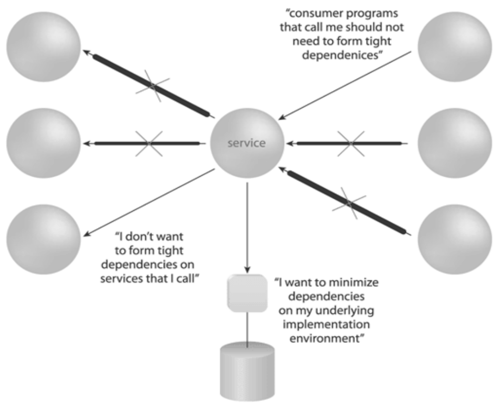
Source: Integraph Corporation

3. SERVICE ORIENTATION PRINCIPLES

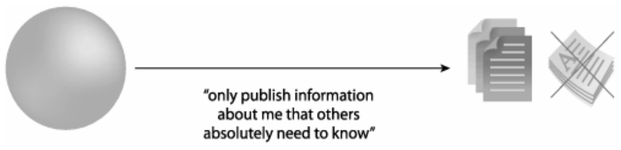
a) **Standardized Service Contract** - Services adhere to a communications agreement, as defined collectively by one or more service-description documents. Services within the same service inventory are in compliance with the same contract design standards.



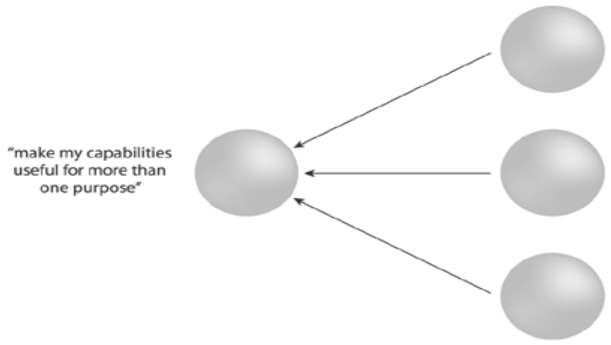
b) **Service Loose Coupling** - Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment. Services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other.


"consumer programs that call me should not need to form tight dependenices"
service
"I don't want to form tight dependencies on services that I call"
"I want to minimize dependencies on my underlying implementation environment"

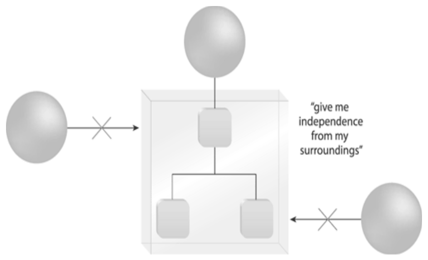
c) **Service Abstraction** - Service abstraction is a design principle that is applied within the service-orientation design paradigm so that the information published in a service contract is limited to what is required to effectively utilize the service. Beyond descriptions in the service contract, services hide logic from the outside world.


"only publish information about me that others absolutely need to know"

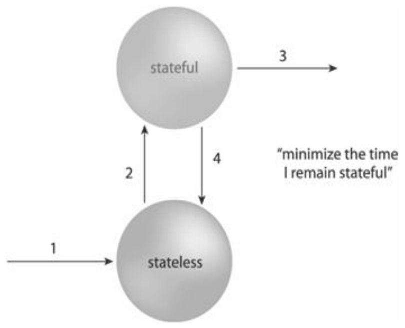
d) **Service Reusability** - Services contain and express agnostic logic and can be positioned as reusable enterprise resources. Logic is divided into services with the intention of promoting reuse.



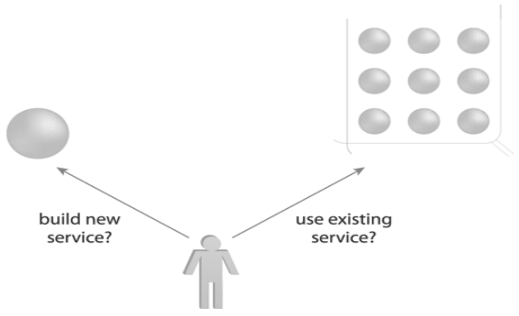
e) **Service Autonomy** - Services exercise a high level of control over their underlying runtime execution environment. Services have control over the logic they encapsulate, from a Design-time and a Run-time perspective.



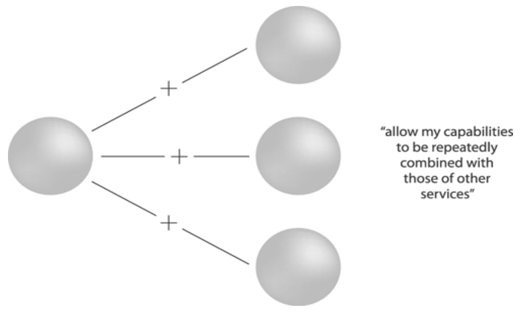
f) **Service Statelessness** - Services minimize resource consumption by deferring the management of state information when necessary.



g) **Service Discoverability** - Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted.

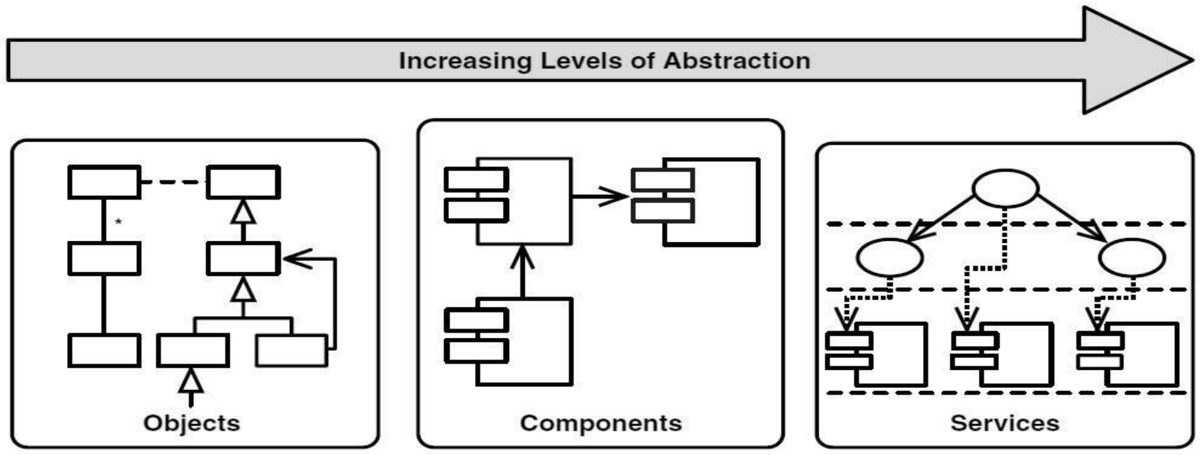


h) **Service Composability** - Services are effective composition participants, regardless of the size and complexity of the composition.


"allow my capabilities to be repeatedly combined with those of other services"

4. FUNDAMENTAL BUILDING BLOCKS OF SOA

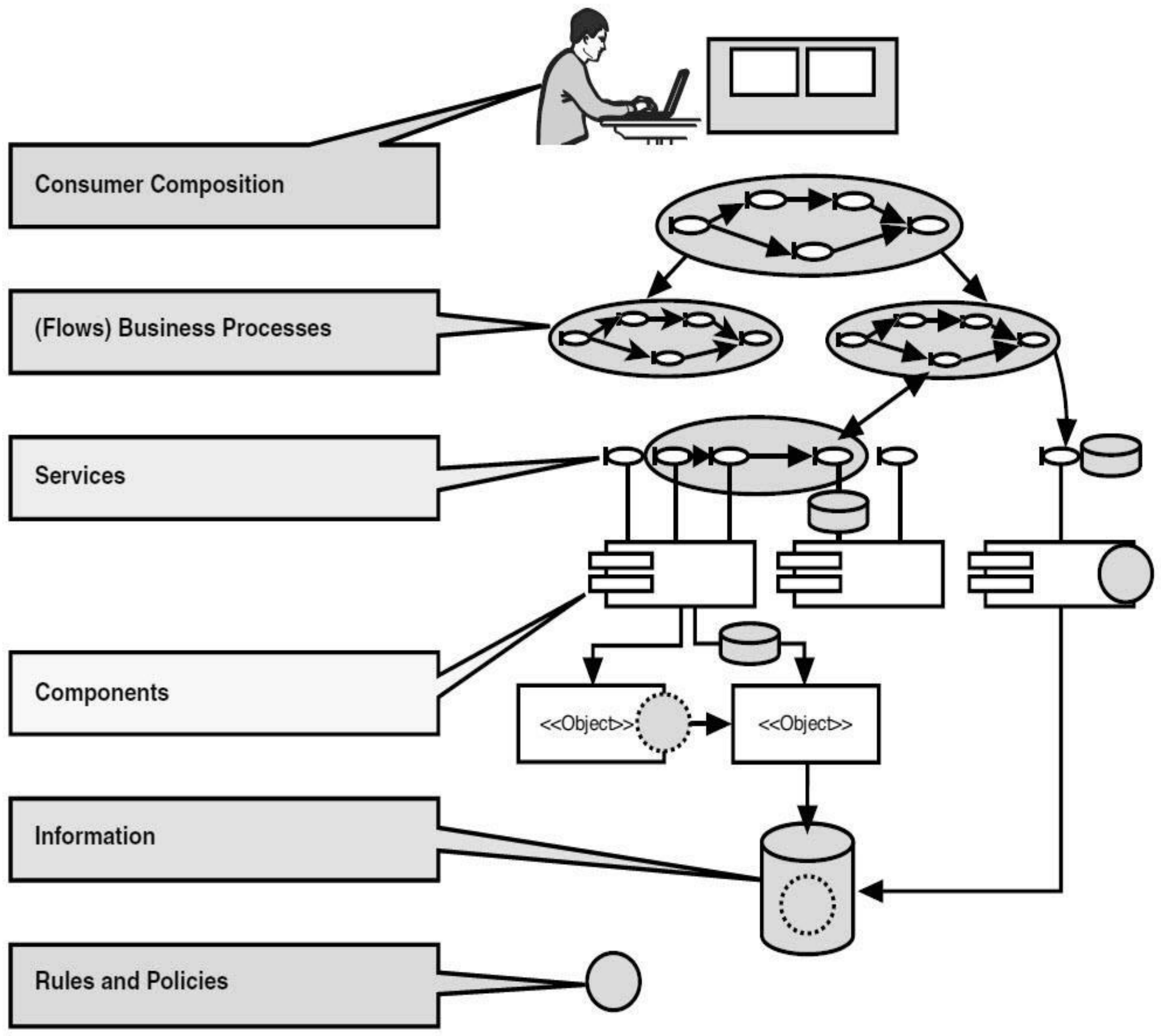
Service is the most basic construct or building block of SOA. Software engineering over the years has evolved from procedural to structured programming to object-oriented programming to component-based development and now to service oriented. Figure below illustrates the different levels of abstraction from objects to services. Each evolution of abstraction builds on the previous, and SOA grasps the best practices of item and segment advancement.



Source: http://www.informit.com/articles/article.aspx?p=1663690

5. TOP-DOWN VIEW OF SOA CONSTRUCTS

SOA comprises the following constructs, as illustrated in the figure below consumer, business processes, services, components, information, rules, and policies. Buyers permit summon or creation of services at the shopper layer through social programming, business forms, or different frameworks. Business methods speak to the streams of exercises needed to finish a business process; they are organizations of services focused to attain to business objectives. Services are the principle organizing component needed by an administration shopper and are given by the administration supplier. Services offer usefulness and nature of administration, both of which are externalized inside administration portrayals and arrangements. Services can be made out of different services, along these lines making them composite services. Parts acknowledge not just the usefulness of the services they uncover additionally guarantee their nature of administration. Data streams between the layers (for instance, buyer, process, and administration) and inside a layer. In conclusion, standards and strategies exist for services, parts, and streams.



Source: http://www.informit.com/articles/article.aspx?p=1663690

6. SOA MYTHS AND FACTS

There are several myths associated with SOA which are very important to understand before digging deeper into it. The table below describes some of the top myths surrounding SOA and the facts to help debunk them.

Myth	**Fact**
SOA is a technology	SOA is a configuration logic autonomous of any merchant, item, and innovation or industry pattern. No seller will ever offer a "complete" SOA "stack" on the grounds that SOA needs differ starting with one association then onto the next. Buying your SOA framework from a solitary seller overcomes the reason for putting resources into SOA.
SOAs require Web Services	SOAs may be realized via Web services but Web services are not necessarily required to implement SOA
SOA is new and revolutionary	EDI, CORBA and DCOM were conceptual examples of SO
SOA ensures the alignment of IT and business	SOA is not a methodology
A SOA Reference Architecture reduces implementation risk	SOAs are like snowflakes – no two are the same. A SOA Reference Architecture may not necessarily provide the best solution for your organization
SOA requires a complete technology and business processes overhaul	SOA should be incremental and built upon your current investments
We need to build a SOA	SOA is a means, not an end

7. WEB SERVICES

A web Service is a product application distinguished by a URI, whose interfaces and tying are fit for being characterized, depicted and found by XML curios and backings direct associations with other programming applications utilizing XML based messages by means of Internet-based conventions.

A Web Service is a product framework intended to bolster interoperable machine-to-machine communication over a system. It has an interface portrayed in a machine-processable organization (particularly WSDL). Different frameworks connect with the Web benefit in a way endorsed by its depiction utilizing SOAP, ordinarily passed on utilizing HTTP with a XML serialization in conjunction with other Web-related gauges.

"Web Services is the innovation that takes into consideration distinctive applications from diverse sources to speak with one another without lengthy custom coding, and in light of the fact that all correspondence is in XML, Web services are not fixing to any one working framework or programming dialect." - Webopedia.

Web Services are basically utilized as a methods for organizations to speak with one another and with customers. Web Services are naturally appropriated instead of the Client/server applications that are essentially information driven in nature. Computerized assets got to through the Internet. Web services are programming controlled assets or practical segments whose abilities can be gotten to at a web URI. Norms based web services use XML to collaborate with one another, which permits them to connection up on interest utilizing free coupling.

8. SERVICE IMPLEMENTATION

The most common type of implementation is Services as Web Services.

- A Web Service is an example of an SOA with a well-defined set of implementation choices. In general, the technology choices are SOAP and the Web Service Definition Language (WSDL); both XML-based. WSDL describes the interface (the "contract"), while SOAP describes the data that is transferred. Because of the platform-neutral nature of XML, SOAP and WSDL, Java is a popular choice for web-service implementation due to its OS-neutrality.
- Web-service systems are an improvement of client/server systems and proprietary object models such as CORBA or COM, because they are standardized and free of many platform constraints. Additionally, the standards, languages and protocols typically used to implement web services helps systems built around them to scale better.

There are two types of Web Services. They are-

a. SOAP
b. RESTful

a) SOAP -

SOAP (Simple Object Access Protocol) which is based in Extensible Markup Language (XML), facilitates communication between application and operating systems. SOAP (Simple Object Access Protocol) is a messaging protocol that allows programs that run on disparate operating systems (such as

Windows and Linux) to communicate using Hypertext Transfer Protocol(HTTP) and its Extensible Markup Language (XML). SOAP Web Services are standard-based and supported by almost every software platform: They rely heavily in XML and have support for transactions, security, asynchronous messages and many other issues. It’s a pretty big and complicated standard, but covers almost every messaging situation.

✓ ***Soap Services are appropriate in this scenarios:***

- **Asynchronous processing and invocation;** if your application needs a guaranteed level of reliability and security then SOAP 1.2 offers additional standards to ensure this type of operation.
- **Formal contracts;** if both sides (provider and consumer) have to agree on the exchange format then SOAP 1.2 gives the rigid specifications for this type of interaction.
- **Stateful operations**; : For example, you store information/data on a request and use that stored data on the next one

b) RESTful –

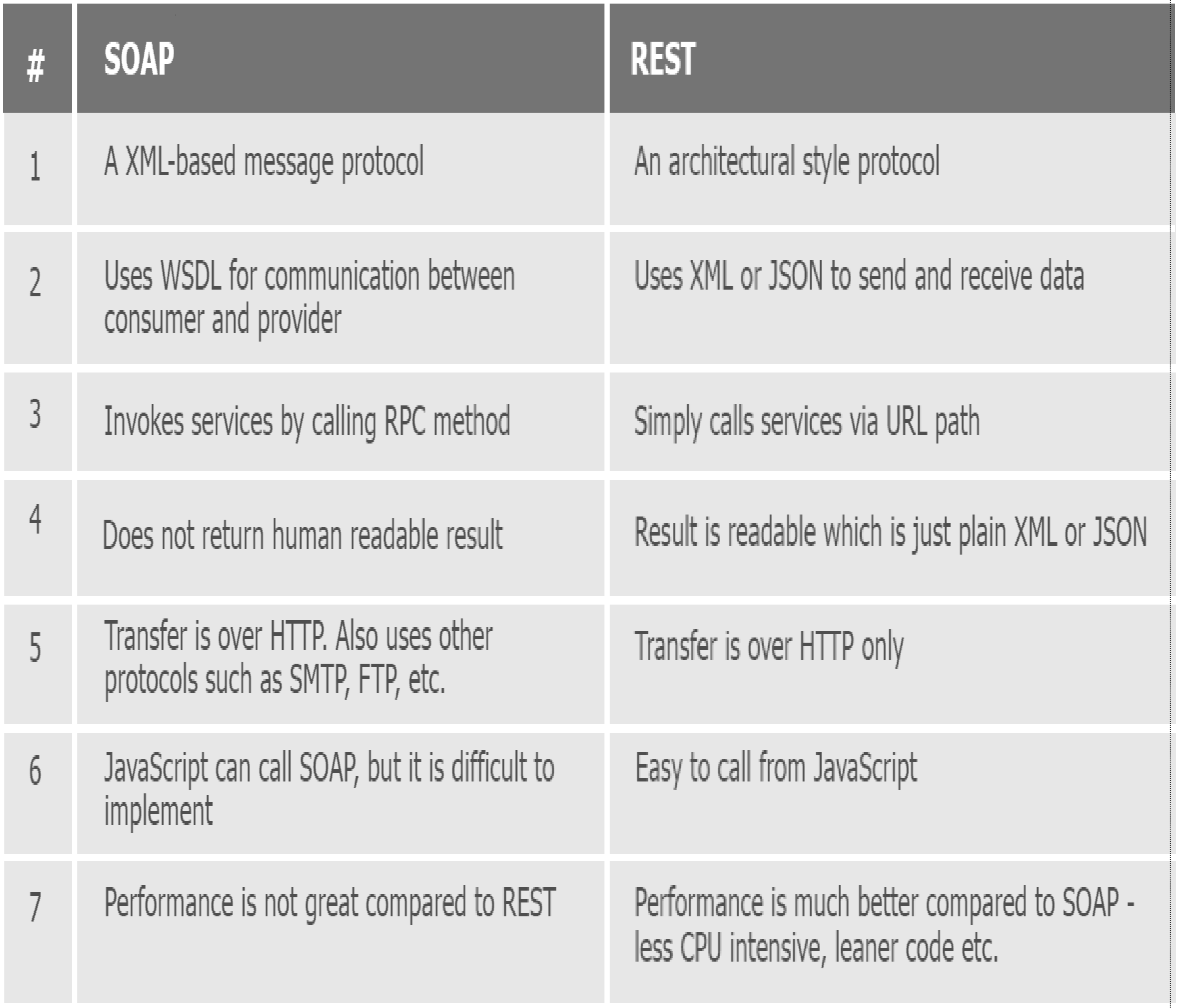
The acronym REST stands for **Representational State Transfer**; this basically means that each unique URL is a representation of some object. A very simplistic implementation of REST *could* use the following CRUD mapping:

- Create -> Post
- Read -> Get
- Update -> Put
- Delete -> Delete

✓ ***RESTful Services are appropriate in this scenarios:***

- **Limited bandwidth and resources;** remember the return structure is really in any format (developer defined). Plus, any browser can be used because the REST approach uses the standard *GET*, *PUT*, *POST*, and *DELETE* verbs. Again, remember that REST can also use the *XMLHttpRequest* object that most modern browsers support today, which adds an extra bonus of AJAX.
- **Totally stateless operations;** if an operation needs to be continued, then REST is not the best approach and SOAP may fit it better. However, if you need stateless CRUD (Create, Read, Update, and Delete) operations, then REST is it.
- **Caching situations;** if the information can be cached because of the totally stateless operation of the REST approach, this is perfect.

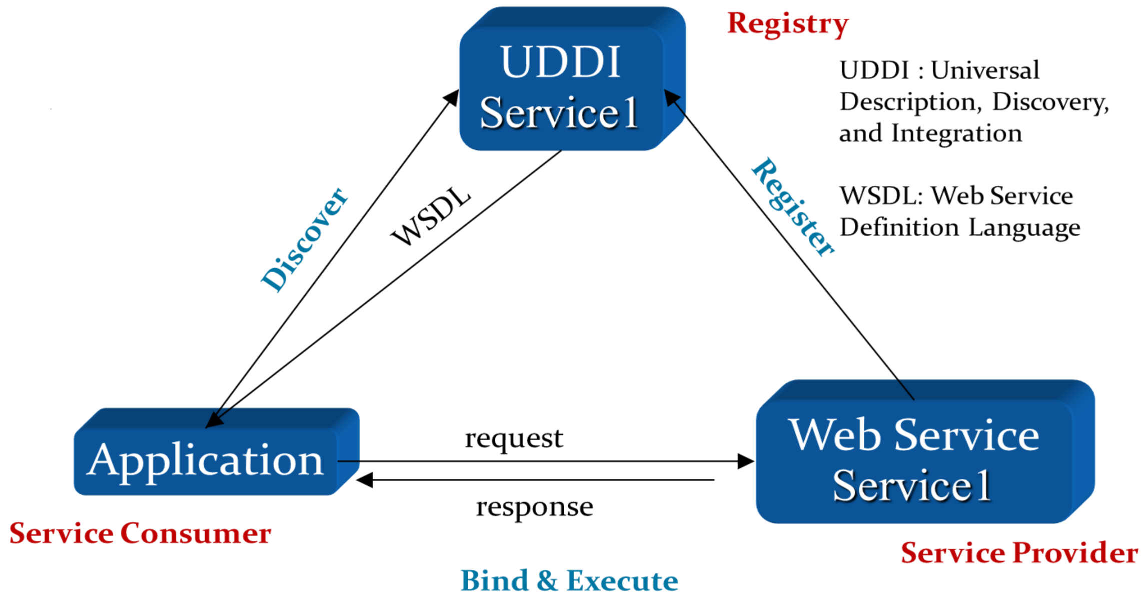
9. SOAP vs. RESTful



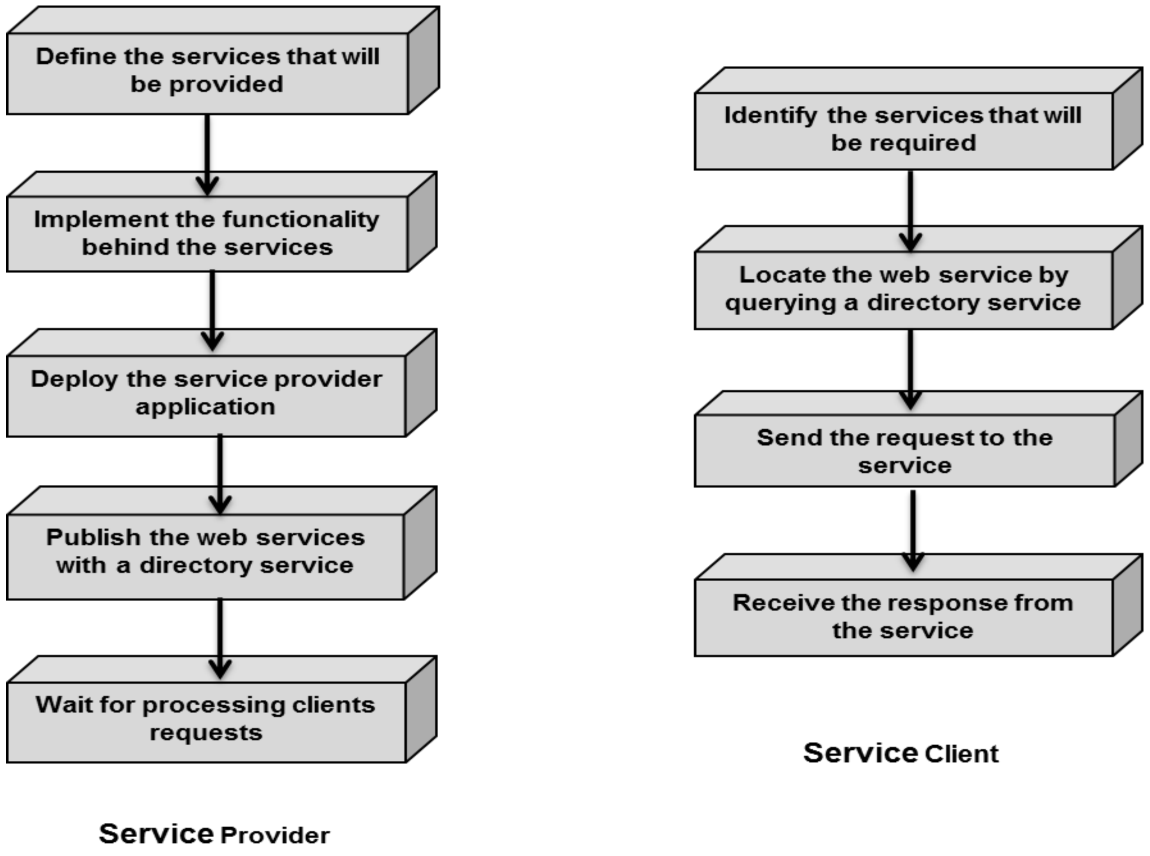
#	SOAP	REST
1	A XML-based message protocol	An architectural style protocol
2	Uses WSDL for communication between consumer and provider	Uses XML or JSON to send and receive data
3	Invokes services by calling RPC method	Simply calls services via URL path
4	Does not return human readable result	Result is readable which is just plain XML or JSON
5	Transfer is over HTTP. Also uses other protocols such as SMTP, FTP, etc.	Transfer is over HTTP only
6	JavaScript can call SOAP, but it is difficult to implement	Easy to call from JavaScript
7	Performance is not great compared to REST	Performance is much better compared to SOAP - less CPU intensive, leaner code etc.

Source: http://www.rapidvaluesolutions.com/web-services-data-transfer-frameworks-for-mobile-enabling-applications/

10.WEB SERVICE INTERACTION

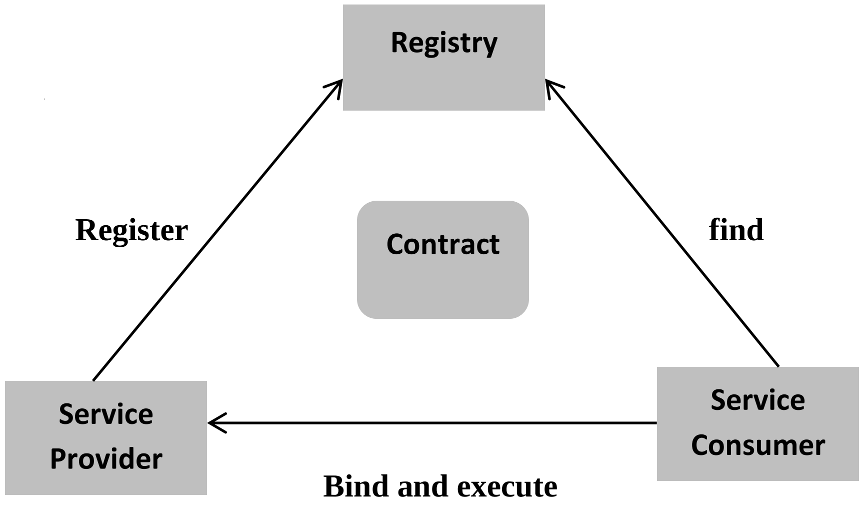


11._STEPS TO CREATE A WEB SERVICE


Define the services that will be provided
Implement the functionality behind the services
Deploy the service provider application
Publish the web services with a directory service
Wait for processing clients requests
Service Provider
Identify the services that will be required
Locate the web service by querying a directory service
Send the request to the service
Receive the response from the service
Service Client

12.WEB SERVICE ROLES

There are three major roles within the web service architecture:



- **Service Provider**
  This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

- **Service Requestor/ Consumer**
  This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

- **Service Registry**
  This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.

13.USAGE EXAMPLES

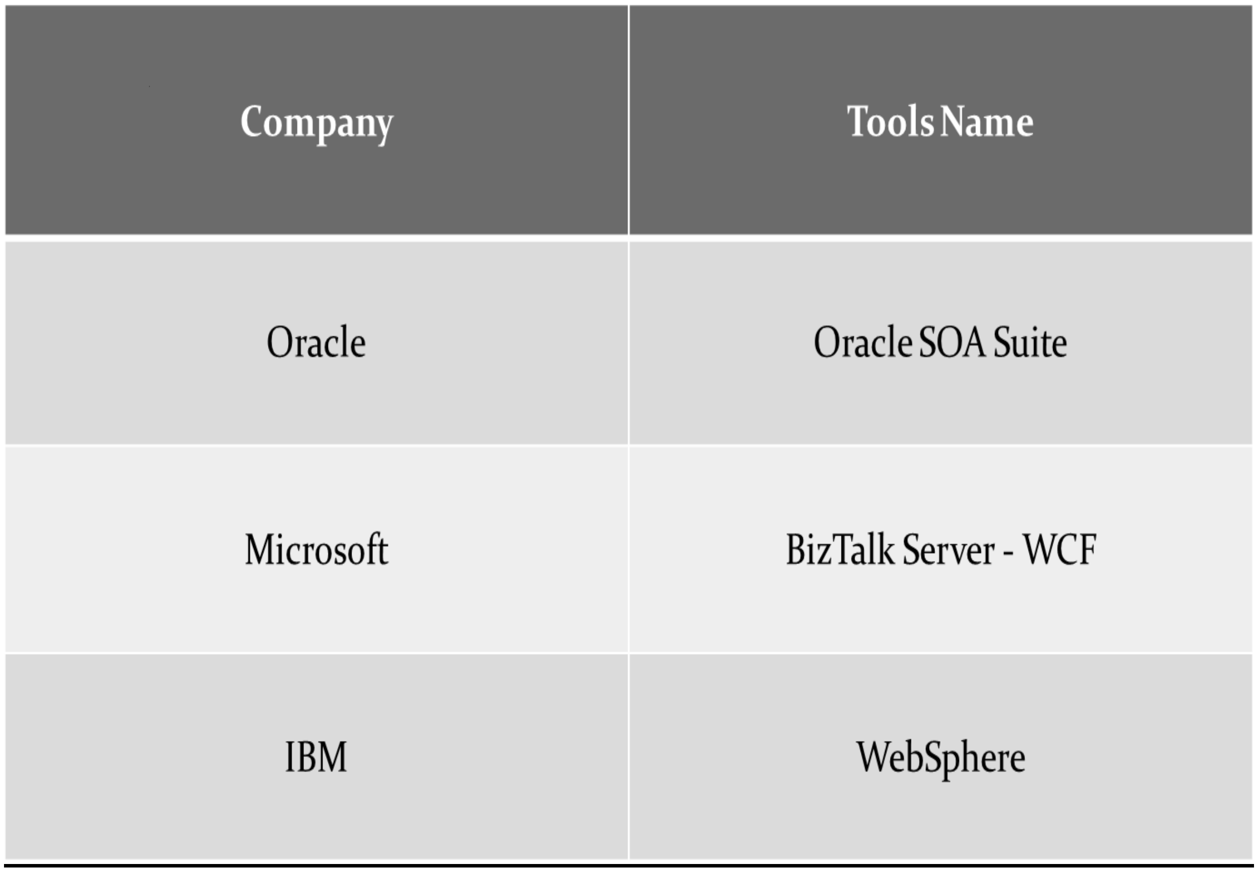
- **Who's using REST?**

All of Yahoo's web services use REST, including Flickr, del.icio.us API uses it, pubsub, bloglines, technorati, and both eBay, and Amazon have web services for both REST and SOAP.

- **Who's using SOAP?**

Google seems to be consistent in implementing their web services to use SOAP, with the exception of Blogger, which uses XML-RPC. You will find SOAP web services in lots of enterprise software as well.

14. SOA TOOLS



Company	Tools Name
Oracle	Oracle SOA Suite
Microsoft	BizTalk Server - WCF
IBM	WebSphere

15. DEMO WEB SERVICES CREATED USING RESTFUL AND SOAP

1) Yahoo Stock Service – Java, RESTful
2) Yahoo Weather Service - .Net, RESTful
3) Yelp 10Restaurents – Java, SOAP
4) Twitter Service – Java, SOAP

REFERENCES

http://searchsoa.techtarget.com/definition/service-oriented-architecture

https://msdn.microsoft.com/en-us/library/bb833022.aspx

https://msdn.microsoft.com/en-us/library/bb833022.aspx

https://msdn.microsoft.com/en-us/library/bb833022.aspx

http://frogslayer.com/expertise/technologies/soa-and-web-services/

http://www.soa.com/solutions/faqs/

http://www.tridens.si/expertise/soa/

http://hyperty.com/portfolio/2008/04/04/vivat-soa/

http://searchitchannel.techtarget.com/feature/Goals-of-SOA-Integration-and-interoperability

http://en.wikipedia.org/wiki/Service-oriented_architecture

http://www.oasis-opencsa.org/sca

http://www.drdobbs.com/web-development/soa-web-services-and-restful-systems/199902676

http://searchsoa.techtarget.com/definition/SOAP

"Business Component Factory", Peter Herzum and Oliver Sims, Wiley, 1999

"Enabling "Real World" SOA through the Microsoft Platform", A Microsoft White Paper, December 2006. Available at http://www.microsoft.com/biztalk/solutions/soa/whitepaper.mspx

http://ptgmedia.pearsoncmg.com/imprint_downloads/informit/promotions/LearnSOA/SOA_eBook-InformIT.pdf

https://msdn.microsoft.com/en-us/library/bb833022.aspx

http://www.developer.com/services/article.php/1010451/Service-Oriented-Architecture-Introduction-Part-1.htm

http://searchsoa.techtarget.com/answer/The-evolution-of-SOA

http://www.tutorialspoint.com/webservices/what_are_web_services.htm

http://searchsoa.techtarget.com/tip/REST-vs-SOAP-How-to-choose-the-best-Web-service

Erl, T.: Service-Oriented Architecture (SOA): Concepts, Technology, and Design.
Prentice Hall PTR, Upper Saddle River (2005)

Fielding, R., Taylor, R.N.: Principled design of the modern Web architecture. ACM
Press New York, NY, USA (2000) 407-416

Balzer, S., Liebig, T., Wagner, M.: Pitfalls of OWL-S: a practical semantic web use
case. ICSOC'04. ACM Press, New York (2004) 289-298

Dietz, J.L.G.: Enterprise Ontology: Theory and Methodology. Springer, New York (2006)

Grigori, D., Corrales, J.C., Bouzeghoub, M.: Behavioral matchmaking for service retrieval, ICWS'06. IEEE Computer Society Press, Los Alamitos, CA. (2006) 145-152

Shirky, C.: Planning for Web Services: Obstacles and Opportunities. O'Reilly Media, Sebastopol, CA. (2002)

Wiegers, K.E.: Software Requirements. 2nd edn. Microsoft Press, Redmond, Washington (2003)