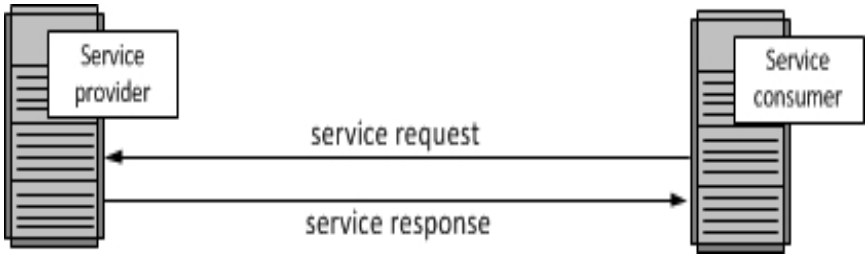
A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed. Service-oriented architectures are not a new thing. The first service-oriented architecture for many people in the past was with the use DCOM or Object Request Brokers (ORBs) based on the CORBA specification.
A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services.

The technology of Web Services is the most likely connection technology of service-oriented architectures. The following figure illustrates a basic service-oriented architecture. It shows a service consumer at the right sending a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider.



What is web Service?
A Web service is a method of communications between two electronic devices over the World Wide Web. It is a software function provided at a network address over the web with the service always on as in the concept of utility computing(*Utility computing is the packaging of computing resources, such as computation, storage and services, as a metered service*).
"Web Services are a way to send and receive information between remote programs."

According to WikiPedia "Web Services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services"
.
There are many advantages of using webservices. Two obvious uses have been mentioned in many of the tutorials. One is the communication between different machines/applications. The next one is to develop reusable application components. Other main advantages is that it can use SOAP messaging which is much more efficient and less bandwidth costly. Another advantage is that Service Transport, XML Messaging, Service Description and Service Discovery layers use a standard protocol witch make it easy for developers to develop functionality irrespective of programming languages.

Benefits of using Web Services:

- Exposing the existing function on to network: A Web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. So, Web Services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.
- Connecting Different Applications ie Interoperability: Web Services allows different applications to talk to each other and share data and services among themselves. Other applications can also use the services of the web services. For example VB or .NET application can talk to java web services and vice versa. So, Web services is used to make the application platform and technology independent.
- Standardized Protocol: Web Services uses standardized industry standard protocol for the

communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) uses the well defined protocol in the Web Services protocol stack. This standardization of protocol stack gives the business many advantages like wide range of choices, reduction in the cost due to competition and increase in the quality.

- Low Cost of communication: We can use your existing low cost internet for implementing Web Services. This solution is much less costly compared to proprietary solutions like EDI/B2B. Web Services can also be implemented on other reliable transport mechanisms like FTP, REST, SOAP, etc.

Before the concept of REST, web services were understood as follows:
Web services describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Used primarily as a means for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall. Unlike traditional client/server models, such as a Web server/Web page system, Web services do not provide the user with a GUI. Web services instead share business logic, data and processes through a programmatic interface across a network. The applications interface, not the users. Developers can then add the Web service to a GUI (such as a Web page or an executable program) to offer specific functionality to users. Web services allow different applications from different sources to communicate with each other without time-consuming custom coding, and because all communication is in XML, Web services are not tied to any one operating system or programming language. For example, Java can talk with Perl, Windows applications can talk with UNIX applications.

Web services' distributed computing model allows application-to-application communication. For example, one purchase-and-ordering application could communicate to an inventory application that specific items need to be reordered. Because of this level of application integration, Web services have grown in popularity and are beginning to improve business processes. In fact, some even call Web services the next evolution of the Web.

Web Services Technology
Web services are built on several technologies that work in conjunction with emerging standards to ensure security and manageability, and to make certain that Web services can be combined to work independent of a vendor. The term Web service describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone.
XML
Short for Extensible Markup Language, a specification developed by the W3C. XML is a pared-down version of SGML, designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.
SOAP
Short for Simple Object Access Protocol, a lightweight XML-based messaging protocol used to encode the information in Web service request and response messages before sending them over a network. SOAP messages are independent of any operating system or protocol and may be transported using a variety of Internet protocols, including SMTP, MIME, and HTTP.
WSDL
Short for Web Services Description Language, an XML-formatted language used to describe a Web service's capabilities as collections of communication endpoints capable of exchanging messages. WSDL is an integral part of UDDI, an XML-based worldwide business registry. WSDL is the language that UDDI uses. WSDL was developed jointly by Microsoft and IBM.
UDDI

Short for Universal Description, Discovery and Integration. It is a Web-based distributed directory that enables businesses to list themselves on the Internet and discover each other, similar to a traditional phone book's yellow and white pages.
XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Used primarily as a means for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall.

Web Security and Security Issues

Security has become a hot topic for Web services. Because it is based on program-to-program interactions as opposed to human-to-program interaction, it is important for Web service security to address topics such as access control, authentication, data integrity and privacy. Today the most common security scheme is SSL (Secure Sockets Layer), but when it comes to Web services there are limitations with SSL. The Web service technology has been moving towards different XML-based security schemes for Web services. Some of the XML-based securities include the following:

- **XML digital signature**

The XML Signature specification is a joint effort of W3C and IETF. XML Signatures provide integrity, message authentication and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere.

- **XML Encryption**

W3C's XML Encryption specification addresses the issue of data confidentiality using encryption techniques. Encrypted data is wrapped inside XML tags defined by the XML Encryption specification.

- **XKMS (XML Key Management Specification)**

The XML Key Management Specification (XKMS) comprises two parts ? the XML Key Information Service Specification (X-KISS) and the XML Key Registration Service Specification (X-KRSS). The X-KISS specification defines a protocol for a Trust service that resolves public key information contained in XML-SIGelements. The X-KISS protocol allows a client of such a service to delegate part or all of the tasks required to process elements. The X-KRSS specification defines a protocol for a web service that accepts registration of public key information. Once registered, the public key may be used in conjunction with other web services including X-KISS.

- **SAML (Secure Assertion Markup Language)**

SAML is an XML-based framework for communicating user authentication, entitlement and attribute information. As its name suggests, SAML allows business entities to make assertions regarding the identity, attributes, and entitlements of a subject (an entity that is often a human user) to other entities, such as a partner company or another enterprise application. The OASIS Security Services Technical Committee is in charge of defining, enhancing, and maintaining the specifications that define SAML.

- **WS-Security (Web Services Security)**

Security Assertion Markup Language (SAML) from OASIS provides a means for partner applications to share user authentication and authorization information. This is essentially the single sign-on (SSO) feature being offered by all major vendors in their e-commerce products. In the absence of any standard protocol on sharing authentication information, vendors normally use cookies in HTTP communication to implement SSO. With the advent of SAML, this same data can be wrapped inside XML in a standard way, so that cookies are not needed and interoperable SSO can be achieved.

- **ebXML Message Service**

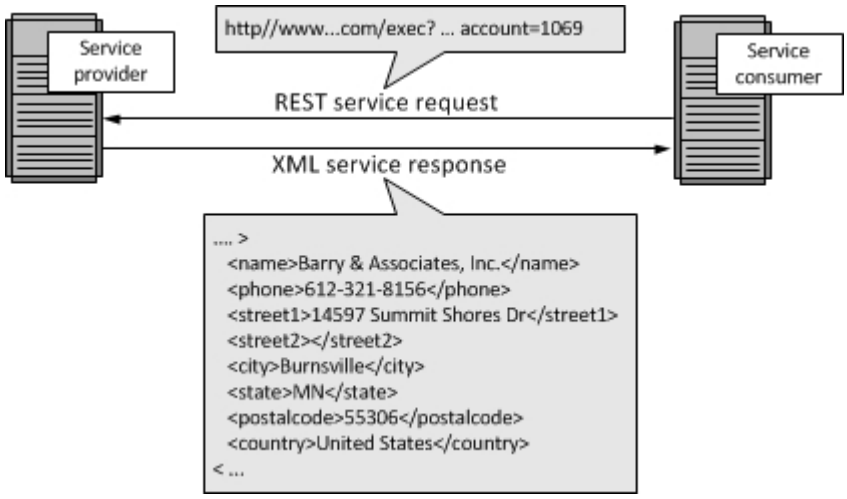
The OASIS ebXML Message Service defines the message enveloping and header document schema used to transfer ebXML messages over a communications protocol such as HTTP or SMTP and the behavior of software sending and receiving ebXML messages.

Who Is Using Web Services?

(adapted from InternetNews.com)
Perhaps the best example of the growth of Web services is eBay. The online auction king has been

aggressively developing its Web services platform by extending application programming interfaces that essentially turn its Web site into a platform.
The auction site's developer section gives soup-to-nuts information about deploying its eBay API. "With the eBay API, you communicate directly with the eBay database in XML format. By using the API, your application can provide a custom interface, functionality and specialized operations not otherwise afforded by the eBay interface." Since 1999, eBay has offered APIs and now offers more than 100 Web services calls available to developers to build applications that can connect to those services. They include pricing information, buy-it-now features, and payment options through its PayPal subsidiary. The growth and use of APIs across the Web illustrate how rapidly Web services are spreading, even as technical issues such as security and authentication are worked out by standards bodies.

REST = “Representational State Transfer”



- It is a software architectural style for distributed hypermedia systems. This style was created by Roy Fielding, and described in his dissertation from UC Irvine, by examining what made the Web such a successful technology. Due to popularity of REST, Roy Fielding has the distinction of having one of the most widely read Ph.D. dissertations
- Is not a standard but it does use standards. It uses HTTP, URI, XML.
- REST puts “Web” back into “Web Services”

- The Internet is a REST system
- In REST architecture the server provides and controls the access to the resources and REST client can access and modify the resources.
- REST support all the common http operations and resources are accessed by URI
- REST allow that resources have different representational styles such as XML,JSON,etc., Client access the these representation by using URI.
- Each URL on the server represents a resource; either a collection resource or an element resource. A collection resource would be available at a URL like http://restful.ex/items/ which would be a representation of a list of items. A element resource would be available at a URL like http://restful.ex/items/2 which would be a representation of a single item, identified by 2.

- Different HTTP methods are used for different CRUD operations:

a GET is a read operation
a PUT is a write/modify operation
a POST is a create/new operation
a DELETE is a remove/delete operation.

- State (or rather, client context) is not stored on the server-side; all state is in the representations passed back and forth by the client's requests and the server's responses.

Note: An identical request on an idempotent method causes at most one state change in the receiving system no matter how many times it is invoked.

REST Design Principles

1. **Stateless Client/Server Protocol**

- Each message in the protocol contains all the information needed by the receiver to understand and/or process it. This constraint attempts to "keep things simple" and avoid needless complexity

2. **Set of Uniquely Addressable Resources**

- "Everything is a Resource" in a RESTful system
- Requires universal syntax for resource identification (e.g. URI)

3. **Set of Well-Defined Operations**

- that can be applied to all resources
- In context of HTTP, the primary methods are: POST, GET, PUT, DELETE
- these are similar (but not exactly) to the database notion of CRUD (Create, Read, Update, Delete)

4. **The use of Hypermedia both for Application Information and State Transitions**

- Resources are typically stored in a structured data format that supports hypermedia links, such as XHTML or XML

Web as embodiment of REST

- Stateless Protocol: HTTP
- Uniquely Addressable Resources: URIs
- Well-Defined Operations: HTTP Methods defined in HTTP 1.1 spec
- Use of Hypermedia: HTML is the primary Web format

But it is important to note that while the Web is ONE embodiment of these principles, it is NOT the ONLY one, we can create RESTful systems using other protocols, methods, resources, and data formats but those new entities must conform to the previous four principles.

Are These Operations Enough?

Four (main) methods for ALL resources?

- Is that enough?
- What about collections?
  - How do I list the members of a collection?
  - How do I find resources?

Answer: Collections and Search Results are viewed as “just another resource”

- They have their own unique URLs and the standard methods can then be applied to them just like any other resource
  - Example: GET /users — Get resource listing all users
  - Example: GET /search/DNC — Get results that match keyword DNC

REST + URI

- REST puts the focus back into a URI rather than obscuring it behind an API

http://some-store.com/categories/

http://some-store.com/products/widget

http://some-store.com/search/gadgets

RESTful Web services depend in large part on the URI to decide what to do. For example, in a "GET" request, the URI path typically contains the primary key value (or some other identifier) of the entity to retrieve. For instance, "http://www.example.org/service/entityname/76" would retrieve the entity with a name of "entityname" and a primary key value of 76. With REST Web services, the URI is not just a means of accessing the service, but a way of controlling it and signaling your needs.

REST vs. RPC

Useful to compare these approaches as RPC underlies all of the middleware systems along with SOAP-based web services. In RPC systems, the design emphasis is on verbs

- What operations can I invoke on a system? getUser(), addUser(), removeUser(), updateUser(), getLocation(), updateLocation(), listUsers(), listLocations(), etc.

In REST systems, the design emphasis is on nouns: e.g. User, Location. In REST, you would define XML representations for these resources and then apply the standard methods to them.

Example

```
<user>
        <name>Jane</name>
        <gender>female</gender>
        <location href="http://www.example.org/locations/us/ny/new_york_city">
                                                New York City, NY, USA</location>
</user>
```

This documentation is a representation used for the User resource. It might live athttp://www.example.org/users/jane/
If a user needs information about Jane, they GET this resource. If they need to modify it, they GET it, modify it, and PUT it back. The href to the Location resource allows savvy clients to gain access to its information with another simple GET request.

Real Life Examples: del.icio.us

- Most popular social bookmarking utility i.e. http://del.icio.us
- A property of Yahoo!

Wicked simple REST API http://del.icio.us/help/api/

The del.icio.us API

- Update
  https://api.del.icio.us/v1/posts/update
- Tags
  https://api.del.icio.us/v1/tags/get
  https://api.del.icio.us/v1/tags/rename

- Simple API = Simple Documentation
  http://del.icio.us/help/api
  Uses HTTP Authentication
  Still under development

- Posts
  https://api.del.icio.us/v1/posts/get
  https://api.del.icio.us/v1/posts/recent
  https://api.del.icio.us/v1/posts/all
  https://api.del.icio.us/v1/posts/dates
  https://api.del.icio.us/v1/posts/add
  https://api.del.icio.us/v1/posts/delete

- Bundles
  https://api.del.icio.us/v1/tags/bundles/all
  https://api.del.icio.us/v1/tags/bundles/set
  https://api.del.icio.us/v1/tags/bundles/delete

References:

http://www.javatutorialscorner.com/2013/08/rest-representational-state-transfer.html
http://www.tutorialspoint.com/webservices/why_web_services.htm
http://www.webopedia.com/DidYouKnow/Computer_Science/2005/web_services.asp
http://en.wikipedia.org/
http://www.techrepublic.com/blog/10-things/10-things-you-should-do-to-write-effective-restful-web-services/
http://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html
http://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html
Introduction to REST: Kenneth M. Anderson,University of Colorado, Boulder