

Future of Web Services



Author: Vincent. T. P

Contact: TVincent@blr.Cognizant.com

Date: August 2002

Introduction

Web services are predicted to be the latest technological change that will revolutionize business. Technology companies and visionaries have started their rhetoric. An entirely new era appears to be emerging where anyone can publish their services using standard Internet protocols and consumers around the world can easily combine these services in any fashion to provide higher order services. These service network chains will solve every business problem in the world and in the process generating revenues to everyone involved in the chain.

Is this a utopian idea or something that can really happen? Though the technology for implementing web services is available and everybody seems to be doing something with it, does it have a future at all? From object-oriented programming, components, middleware, dot com et al, we have heard many similar promises.

When the whole world is moving in one direction, the only alternate for a skeptic is to look back into the past, find some reference points from there and analyze the sanity of the current frenzy. This becomes tough when the change is in the field of technology, as it does not have any predecessors. Every change is unique and can alter the whole history. Still, there will be some data points that we can gather from our experience and extrapolate it to the future. Can we predict the future of web services by comparing it with the real world service interactions?

As evident from real world services, we can expect only a certain type of service relationship to form sustainable business models. This report attempts to predict the nature of the successful web services by comparing it with the real world web services. We also try to extrapolate the technology future of web services using the same type of extrapolation.

This article assumes a preliminary knowledge of the basic infrastructure of web services like SOAP, XML, UDDI, WSDL etc. and similar web services related technologies. The attempt is not to explain or review any of these technologies. As we all know, practically any software application can be exposed as a set of services and described using these emerging technology standards. The technologies also allow these services to be combined together to form more complex services. Unlike applications being the end in itself with a user interface, the data and functionalities of the applications are exposed for other consumers to integrate. Also, the process of identification of the services and using them are simplified by using the ubiquitous Internet protocol. This ease and flexibility definitely opens up many sources of

distinct branches of business like exposing the existing functionalities through web services, creating brand new web services, creating infrastructure for web services based interactions, handling the revenue generations and sharing mechanisms etc. However, the real challenge is in identifying the philosophy behind the successful web services – by identifying the characteristics of those that have an higher chance of revenue generation and identifying those that are doomed to fail.

This analysis is particularly important in the current context. Painful memories of the dot com era still linger and the natural doubt is whether the new wave also will face the same fate.

Separation of Business Model from Technology

There is a growing confusion of the technologies/standards used and the business model of web services. Just because SOAP is used for integration between NT and UNIX boxes in an application, we cannot say that it is an example of a web service. Unfortunately, vendors and software construction groups confuse the use of SOAP/XML/WSDL etc. and the service layer of the architecture as the success of web services. The web services technology of SOAP/XML etc. and the services layer as an architecture concept will find use within the organization for integration between disparate systems. This is a loosely coupled way of connecting the applications.

Many similar technologies namely RPC, DCOM/COM+, CORBA, RMI etc. are available to connect the applications. SOAP is probably the latest addition to this list. Every one of these technologies have some advantages and disadvantages. Clearly SOAP has some unique advantages. But this technology is more microscopic and application level connectivity mechanism than true web services. Why is it different from the macroscopic type of service that spans multiple revenue generating units, which we call as the actual web service? Both may use the same technology. But the initial one can make hard decisions on how different services are tied together. The services need not be atomic or well defined. The integration can be complex. Application developers within the boundaries of single application or organization units can dictate the way the services can be combined. None of these rules can be imposed on an externally provided service. The externally published web services also bring the service level agreement (SLA) and other contractual and commitment issues associated with the security and reliability of the services. It also has to negotiate with other similar providers in that segment and standardize the interface definitions. It has to publicize those interfaces and enable clients to access those services with very little effort.

In short, the web services technology used to glue the existing inter company or business-to-business interactions are much different. It is also far simpler than the actual web services hosted as atomic application units exposed for the whole universe to use and aggregate in an arbitrary manner and fetch revenue for the provider. It is the characteristics of the latter type of services that we are interested in exploring in this report. One of the common confusion that you can expect in future is that many vendors harping on the use of SOAP/XML or related technologies/standards/tools as a success story of web services.

Analysis of real world services

In order to understand the impact of Web services, we need to look at the way the real world services are operating. Let us analyze these different service relationships in detail to identify the major characteristics.

Let us look at some of the real world services scenario. An airline provides transportation services. An airline reservation application provides reservation services. A travel agent provides the services to identify the correct flight based on your budget, time etc. A travel web site allows you to find the best price for a ticket and other personal profile information ranging from your meal, seat, special privileges, mile tracking and many other preferences. A travel agent can also provide the services handling air, hotels and car reservations together. In short, everyday we see a variety of services helping us to live a normal life. We also find that

these services are heavily interconnected. Some of the services are dependent upon other services. Every service provides some value to you. The ticket amount that you give to the agent is split up between the agents, airlines, catering services and everyone makes their profit and exist as viable business models. Can the same scheme work if these physical services are replaced with web services?

Every service in the hierarchy adds some value. Otherwise that service will not be there at all. For example, the extra value addition that the agent provides over air, hotel and car reservation is the value addition provided by ensuring that the reservations satisfy personal preferences, correctness of the time dependencies, best deals etc.

Though in theory at least, you can telephone all the airlines and find out the best deals, you leave it to the agent or a website since your time is precious. You find that the value you get from the service far outweighs the extra charge you pay for the services. You derive this value from the time you saved, the best deals you get, the extra reduction in ticket price that the service provider may be able to provide based on their deals with airlines and so on.

Another important aspect of services is its capability of being combined to provide higher order services. The different services can be related in different ways. Someone can combine various services to provide a unified service with some extra value additions. Like a package tour operator combines the services of Cruise, tourist guides, art programs etc. This creates a master/slave relationship between the provider and the integrator. Since the provider may be integrating too many other services and can provide the services to more than one integrator, this causes a web of relationships. It means every node in the service chain aggregates the services of many other slave services and also exposes it to more than one master services – other aggregators or end users. Like a tourist operator aggregates the services of transportation, accommodation, tour guide etc. and renders it to visitors from another country.

Now that we have understood the basic structure of real world services, let us analyze the characteristics of some of the successful ones.

Let us look at refrigeration as a service. Assume a new service that allow storage of vegetables, fruits, medicines etc. into a locker and sets the temperature as per the need. This service will ensure that everything you had stored will be secure for as long as you want. Assume such a centralized facility comes to your town. Will it replace the need for you to have refrigerator at your home? I think it will never.

Now consider another service where instead of refrigeration, the facility is offering services to keep anything in safe deposit in a much larger storage area that can hold furniture, vehicle, mattresses etc. If you leave the place and want to come back only after a long time, this service allows you to keep all your household furniture without paying the rent for your apartment. This may be a successful business. Why one storage service is making business sense whereas other does not?

In the earlier case, the reason for failure is that we want the items to be available quickly - like grabbing a beer bottle without even shifting your eyes from the TV or taking a medicine right in the middle of night. It is impractical and too time consuming to travel frequently and get the items as and when we want even if it is costing 100 times less. On the other hand, if the item is less frequently accessed (you are leaving the place for few months), it makes business sense to avail the storage capability. So the bottom line is that you take the remote or 3rd party help only for those services that are less frequently accessed or less mission critical.

Sometimes the service can also be rendered locally in a one-to-one manner. Let us take the case of a hair dressing service. It can be rendered either as a service in a saloon that the client has to visit or as a hairdresser visiting the client. Is there a difference between these two? One is a service executed at the premises of the service provider, which is totally different from the same service rendered by having some intelligent agent at the client place. Sometimes the need for this type of service is due to the nature of the service itself - like baby-sitting or the need of the end user. You can find many services to have this type of personalized one-to-one local rendering of the service than mere accessing of the service.

Like eating from a fast food joint vs. a chef coming and preparing the dinner; going to a gym vs. a physical trainer coming and training at home etc. The moment the service is rendered locally using an agent, it brings with it all the security issues you can imagine. How do we trust this person? What are all the access controls we need on that individual? Where can we escalate in case of an issue? What is the guarantee that the person who renders the service will not cheat us later?

Almost all the services have implicit or explicit service level agreements. Few minutes after calling the cab, you get down to the hotel lobby and expect to find the cab there. You order a Pizza and do not consider it a miracle when it is delivered to you in less than 30 minutes. We cannot live with services that do not adhere to service level agreements. It may not be written down always. Implicit or explicit, it exists so that you can focus on important things in life.

Another interesting aspect of real world services is the way revenue split happens. How is the revenue distributed among the chain of service providers? The revenue collected by the front-ending service like tour operator is getting divided across the hierarchy of the sub services, based on the value of the services. At each level there is competition (in most cases) to ensure that one service provider does not charge more than the actual value of the service. Such a fair means of revenue distribution is needed for successful business models.

Cornerstone of the real world service is that there is a great amount of trust in all the service transactions. It is unlikely that someone will not pay for a service after having a nice meal at the restaurant. You are not afraid that the firm who took the fat check as advance for building your house is going to vanish with it. In short, you trust all your service providers and service providers trust you. The reason for this mutual trust is many. You know there are places where you can complain if anything wrong happens. You also know that the other party will have to pay a huge price if they are caught by the iron hand of the law.

But your trust is a local phenomenon. You do not trust someone in another part of the world where your laws do not hold any water and you are in no position to complain. Your trust is the bye product of your country, politics, regional affinities, past knowledge, credibility of the provider etc. It will be very difficult for you to wake up tomorrow morning and start trusting the whole world.

Comparison between real world services and web services

Now let us compare the above observations with the web services. Web services technically allow the same mix and match approach of the real world services. However, some of the major differences that we should keep in mind while comparing the ultimate user of a real world service say a human being with that of the user of the web service say a computer. Please note that the ultimate end user may still be the human being even in web services. But it is another computer that aggregates the services before presenting to the end user of the web services.

- The lowermost denominator, the human being, ultimately limits a real world service. With all the technologies and tools, still the human being has limited mental space, time and energy to do things locally. Even if I can save couple of dollars by doing the airline booking by personally checking all the airlines and all the deals instead of relying on a travel agent, I am never going to do it. But a computer is different. It has the space, processing power and determination to do it. So, by doing a service locally if the machine can save something, that will be the normal way of implementation.
- The processing power and storage capacity of the computer is steadily increasing. So an implementation that we think needs remote help today will very well be implemented locally tomorrow. In a computer with less than 1MB RAM, storing the entire dictionary in the memory may be overkill. A service to access it remotely may not sound that crazy. But with today's type of processors and hard disk capabilities, a local storage of the dictionary will be the most common implementation.

- ❑ There is no trust enforced by the countries or governments in Internet and hence on web services. You have to trust some of the leading players merely by their names and past track record.

Now that we have understood some of the real world services, the differences between them and the web services, let us look at the technology of web services.

Business applications were exposing the functionalities through well-defined interfaces even in the past. Other business applications were built over these exposed functionalities. All the airline reservations are going through centralized reservation systems called CRS and the extra services can be built upon it. So what is it that fundamentally different in web services? The only difference is that while these individual wiring between the backend services were one to one earlier, web services allow a mechanism to publish the services and provide those services to whoever is interested with absolutely no change to the service provider implementation. Also the mechanism for identifying and accessing the services is not proprietary but is based on open standards.

This has some impact on the way businesses expose and use the services. To give another scenario, consider what happens if you want to implement a travel page within your company's intranet so that you can allow the individuals in your company to reserve their travels. Also assume that you have a special deal with one of the travel services website. The older way of implementation of this type of co-branding is to redirect the requests to the travel service website with some special hidden identification. The travel services website would then render the GUI and services according to the client. This works very well when the number of co-brands is very less. But each co-branding requires different pages to be rendered based on the customization logic and also different billing rules based on the deals. In short there is a significant amount of GUI and non-GUI changes happening at the server side when a new co-brand is added. In the new web services model, the travel services website can expose all of its services including price negotiations, billing etc. through XMLs. Do not worry about how many are using these services – as long as they pay for it.

So one of the great advantages of web services is that the provider exposes the services without much knowledge of how it is being used by the end application. Instead of the application being the end in itself, it is amenable to be combined with many other complimentary services using simple standard protocols.

Now let us analyze the key issues that we identified earlier with regard to comparison with real world services from a technology perspective.

- ❑ The service must be granular for easier integration. The more complex the functionality more tight will be the integration. The chances are more that you get locked to a single vendor. The reason why ASP (application service providers) model was not so popular was that it demanded a tighter integration and had taken away the flexibility of mix and match.
- ❑ If you compare the new wave of reusability with the previous wave called components, we will find another trend. Reusability is successful when the component granularity is less (like an encryption component, a visual control etc.) or when the application as a whole is being reused – SAP, Peoplesoft etc. There is a huge middle ground where the components (or reuse in general) are not successful.
- ❑ Service level agreements – Current Internet protocols and hence the web services do not have SLAs built into it. So migrating mission critical applications with predictable response time to web services model may not be easy. The future evolutions may take care of it.
- ❑ Local rendering of the services using the agents. This brings with it whole lot of issues. It is like an application unit that can come from the service provider and run locally on behalf of the server, like an applet downloaded from the server executed in the user's browser. There is no current technology to which I can map this concept. The current mode is where the consumers searches for the services, find it out and the actual execution for the service always happen at the service provider. This means that we have a model to provide services like that of a crèche but not like that of a baby sitter yet.

- The web services is rendered as XML . This is lifeless data. We can foresee that we may not be content with lifeless XMLs for long. We want the programs that manipulate, also to come to the client in some cases. This program may still use the web services from the server to access data and modify itself. But essentially this will act as the proxy that can do the jobs at the client place. The advantages are many - no need for going back and forth for simple manipulations, ensures some service levels even in case of n/w failure etc. This will throw more challenges. We need a piece of code that can work in variety of different platforms. What about the security of that component – Is it .NET or Java or both? How do you ensure that this piece of code can access only certain aspects of the application?
- This means that there will be a need for a local sandbox that can allow the application units to flow to consumer's machine and execute it locally. This is similar to the browser. By having a GUI standard we are able to access many sites that execute within our machine. When we move from a GUI world to a non-GUI services mode, we need a similar local box definition that can search, find and execute the local agents of the application. No equivalent concept is available in web services yet.
- Institutions like governments control the real world web services where established social and security frameworks protect the provider and consumer. This is also associated with bulletproof identification mechanisms in real life of the service provider and the consumer. Both identification and security of web services are improving. But the current levels are almost same as that of Internet, which is not that great for running critical businesses.
- The real world business works with payment and division of it across different service providers. Unless arbitration happens at each level for sharing the revenue, the business model will fail. Current revenue models can be extended to web services but trusting other party is a real pre-condition even before we think of revenue split.

Characteristics of viable web services business models

In the end analysis of viability of the business model, we have to identify how the revenue generation and distribution can happen.

The precondition for a viable business model is that all the web services have to be charged. Only this can create a viable economical model. But there will be too many players offering free web services just to penetrate into the market. Like a software development company that uses its spare resources during idle time to develop web services and places it on their existing website absolutely free for increasing their name and fame. Web services itself will be successful only at the lower granular level (otherwise they will go the same path as the ASP model) and this segment will be full of so many players wooing their customers with free services. My point is that there is a thin line between free services and the services that people will be ready to pay. As the service levels are increased or the dependency between the services are more, there may be few dominant players in the market. But there will be so many short time players offering free web services that can actually break the whole market. Some of the same mistakes of dot com era may be repeated.

Though the concept of the end user arbitrarily mixing and matching the services from different vendors looks very interesting, in reality the basic issues of service level agreements will be forcing people to shift to more reliable services. So there may be some SLA based service providers who are quite different from the free service provider. However, how the SLA can be established to be acceptable across the clients will be another market. This is where the accreditations will come into play. Like SEI CMM level certification for software companies, some industry bodies will emerge to certify the services in terms of its reliability. So some may opt for level 1 and some for level 5 and with wide differences in pricing. Security levels and response times will be defined in most of the high-end web services interactions.

If someone is paying the price for the service for a long period, the next question will be the cost of maintaining the code for doing it locally. If this cost is very less then that may be the preferred approach. So from a long-term perspective, if the cost of the service executable or

source code is less than the amount that will be incurred by the service provider, then local implementation will be the preferred approach.

So from a business standpoint, the business web services that will make money should satisfy the following criteria:

- ❑ The functionality should not be too small that there are many players offering free services or it is easier to implement locally.
- ❑ The functionality should not be too high so that it is almost like a one-to-one wiring between the applications. It makes the number of such implementations very limited unless they are the leading players. It also makes the implementation and change of these implementations to be very costly.
- ❑ The cost of the service per year should be less than the cost of developing it and maintaining it locally over a period of time.
- ❑ The cost of the service per year should be less than the cost of acquiring a similar product from a vendor and integrating it- considering even the maintenance costs.
- ❑ The service should be something more than for which the source code is publicly available as open source or can be easily developed.
- ❑ The data transfer of the service - to or from the consumer application- cannot be too huge since the availability of bandwidth will be precious at any time. As the bandwidth availability is high, so will be the size of the data.
- ❑ The frequency of the service accessed from a single application cannot be too large unless the data itself is very volatile.
- ❑ As the cost of the hardware reduces and more and more processing and storage space is available, the local implementation of the services will be preferred.

Let us identify these identifiable parameters for defining the success of a web service.

Attribute	Comment
Granularity of the Web Service	Smaller granularity means easier interface and more adaptability. Bigger services are complex to reuse.
Dependency between the services	More dependency means you are moving ASP path, which will be eventually sure failure.
Data volume	More volume (from and to the web service) needs more bandwidth usage and more delay in response. These are indications for failure.
Complexity of the application logic	Complexity of the implementation is less means any serious business will try to implement it locally. So a service for implementing string manipulations or interest calculations can never make a business.
Availability of the software for application logic from 3 rd parties - free or at affordable price.	Local implementation will be preferred – reliability of the service, security, response time etc. will be motivation for local implementation if the cost is not so high. So a dictionary service, spell check service etc. will never find an audience.
Is it extending a real world service?	The more useful the actual real world service is, the more the chances of someone actually using it. So a hospital exposing its facilities as web services or a hotel exposing its menu as web services will have more integration future. However the actual revenue recognition will be piggybacking on the real world revenue

	recognition models mostly. So you go to a travel booking web services and they have deals with airlines and you get discount.
Data volatility	If the data that you have is extremely volatile, there will be more chance of use.
Proprietary nature of the data/processing	The end user will be forced to use your data anyway since there are no other options like credit rating, media viewership analysis, consumer market research data etc. It is probable that you already have a service layer in your architecture with APIs. SOAP/XML may be another technology to expose your services. Extra revenue is questionable since whoever wants your data are anyway connected to you.

Analysis of some Web services

As per our analysis we can find that almost all the web services that are quoted as sample web services or available in public service providers, does not make much business sense at all. This includes dictionary service, address book, ATM Locations, Auto Loan Calculator, Quote for the day etc.

Based on the above conditions, the following are examples of some of services that can be successful.

- ❑ Highly volatile data application that the client application is forced to look up to the service provider – like airline reservation, stock quotes etc. Note that your few lines of code is not going to fetch much revenue. It will be the actual keepers of this data (like CRS systems and Nasdaq) themselves providing it as service in addition to the current API models.
- ❑ Services that accesses a small subset from very large volumes of data – like satellite image of a town, weather history of a specific region etc. Client cannot store it locally – at least not yet.
- ❑ Services that provide huge processing, like media analysis. Though the final Rating of a TV program is a number, the data that is collected and analyzed for that is so huge that we cannot replicate it locally. The data collection, cleaning, statistical predictions etc. have been perfected over a period of years and there is no way we can rewrite it. It also changes every day. Another example is credit rating analysis services of companies.
- ❑ Conventional services offered as web services, like a restaurant exposing its reservation and a mobile application provider integrates it with a geographical service that allow you to search for the best restaurant nearby and reserve a place for you while you are driving.
- ❑ Extremely complex applications using proprietary algorithms that are not easy to imitate, like a triangulation algorithm for detecting the position based on angle of arrival of the signals – as the accuracy required is more, the geographical details are more complex than a mere spherical equation.

Conclusion

Though technically SOAP, XML, WSDL etc. will continue to penetrate as technologies for connecting applications in a loosely coupled manner, the use of these technologies for building successful business models is a different ballgame altogether. The successful web service business models need to satisfy too many criteria. In the end analysis, only the existing physical services makes sense even as web services most of the time. There will be too many constraints on pure software based web services that are not built on real world services. The free services available anytime in the market and the lower granularity requirement of the service for its adaptation will work in opposing directions leaving only very few services to be viable as business models. One of the pre-requisite for trust in using web

services will be the need for identification of both the service provider and consumer in a vendor independent manner. This is yet to be agreed upon. Considering the security loopholes in the current Internet infrastructure and products, this is a long way to go. The revenue sharing models need someone aggregating the services and collecting the money and sharing it among others. So there will be a tendency by some giants to consolidate different web services with a common infrastructure framework.

So as a business model, web services will be successful only as a technology for exposing the existing real world services and will not create new business models. In order for it to expose pure software services as viable web services, there are many limitations in the current model that need serious rework in the current technology. It may eventually happen and we are definitely in the right path. But few more technology iterations along with some unified industry agreements on identification, authentication and improvement in all security infrastructure is needed before we can dream of someone writing a piece of code and the whole world accessing it and fetching revenues for the creator. It is a dream that we need to keep as target. But we have not yet reached there yet.