This paper will describe what experience I have with the development of web services from an architectural viewpoint. My daily work is as an IT Consultant who does primary Software Architecture. Currently I’m involved in the development of an intelligent Internet archive, which exposes his interfaces as web services to third party ISV’s.

I worked several years for IBM and another outsourcing company where I had some projects where we had to define the SLA for the application we had developed. In one case we developed the application for one of our Clients and he decided to outsource the application to our data center. The key was to define the SLA for each component in the system beginning from the availability and average response time up to the maximum of users who can access this component. This was very challenging since not all components of the system were placed in the data center. In another case we defined SLA’s in an ASP concept for several customers that uses the same application. The challenge here was to find the highest economic possible SLA and the fitting with the customer’s requirements. Response time and data protection where the key items in such contracts. From the ASP provider side we had to introduce some extensions to the application to handle the accounting for each customer.

With the upcoming of SOAP I found that this could be the glue between loosely coupled systems, which is often the case. When we look at CORBA or RMI we have a too tight coupling between the two applications that have to talk. Not at least SOAP gave us the possibility to send a message, which is readable by humans. One of the key concepts of a web service is that there are no dependencies to the implementation language. To expose such services for public the mechanism to discover the service based on taxonomy and have a service description where the constraints are defined is key. With UDDI and WSDL the first step is done to get the things on their way.

Web service should provide a very high level interface, which wraps at least a not so small part of a business process. When we define a web service as one operation that produces a standalone useful result, we have to think about how we can describe the business constraints around such an operation. If we look at the current WSDL spec, little is available to describe such constraints and UDDI is to open from my point of view.

To use the web service from anyone who provides a suiting interface for me needs more information than that we can currently find in the WSDL descriptions. From the B2B perspective there are sometimes lesser requirements regarding the description of such a service. If I know the partner and trust him there is no big hurdle to use his exposed web service. In a B2C environment interaction with web services are
different. Some of these I use more often, other only once a time. I often don’t know much of the provider and what he does with the data. On the other side the provider of such a web service will charge money for his offerings, but depending who invokes the service and from where there must be different rules applied. Since web service can be invoked asynchronously the customer has to know the average and maximum processing time.

If I have a synchronous interface I can interrupt any processing when I disconnect the caller from the service. By using the asynchronous way I can’t do this so. Once the request is invoked I have no chance to cancel the processing if we follow the above description of a web service. In the ASP environment we defined for each application the average response time and the maximum response time. The maximum response time is more critical since this value can work as a timeout value. As a normal end user I will know if a service can process my request in real time or is it queued for later processing, since I have maybe time constraints on getting the result.

Web service can be chained together, this could be happen dynamically or static, in both cases the successful processing depends on the availability of the called services. If those services expose no SLA I cant make any statement about the reliability of my service. In the case of chained services it could happen that these services have to pass the credentials of the caller to the next service in the chain, since this can be essential what the service can do and what SLA applies. The question that comes up to me is what should be part of message for the operation and do we need a general mechanism to send credential information.

Another problem is the system failure of a web service. In an SLA like agreement I have to describe what will happen when the system crashes after I’ve invoked the service and I expect no result back. Is the request discarded or is it saved for later processing. The problems when working with asynchronous messages are not new, but I cant find currently any schema solve this in a standard way in a web service environment.

In the web services we are currently developing, we have a mix of synchronous and asynchronous services. All asynchronous services can process the request further after a system failure, the need for standard way to express at least one level of SLA is obvious.

SLA agreements in the web service world seem to be a multi layer document. One is defined at development time, one at deployment time and the last one at invocation time. At development time we can define such things as the asynchronous behaviour. In the deployment time layer we can define the availability, max response time and other similar stuff, the invocation layer defines payment, what the service can do with the supplied credentials.

I hope this workshop can help to get a common understanding of the problems when I expose web services and start the work to find a solution.