EMP 5102 – Systems Engineering and Integration

Systems Integration – Web services

Resources

- **Web Services Essentials - Distributed Applications with XML-RPC, SOAP, UDDI & WSDL,** Ethan Cerami, O'Reilly, February 2002
- **A Web Services Primer**, Venu Vasudevan

http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html?page=1

- ***Introduction to Web services architecture***

K. Gottshalk, S. Graham, H. Kreger, and J. Snell, IBM Systems Journal, Vol 41 num. 2, 2002

http://www.research.ibm.com/journal/sj/412/gottschalk.pdf

- ***http://www.w3.org/TR/SOAP***
- ***http://www.w3.org/TR/wsdl***

Introduction

One of the technical challenges of systems integration is data sharing amongst the applications involved in the integrated system, and beyond simple data sharing, functional integration allowing one part of the system to “call” another part and use the result.

Data integration

Data integration can be achieved in a variety of ways:

- Manual data transfer: data is entered manually in each system. Obvious problems include scaling and consistency.
- Automated data transfer: the data transfer is done programmatically, either through regular batches or run as required. This approach should resolve the consistency problems (assuming that there is a clear “data owner”), but:
  - May not scale as required if the transfer process is heavy
  - May not be as real time as required
  - Involve n^2 synchronization process to create/maintain
  - Data integrity is difficult/impossible to ensure

Data integration

- Common data source: the systems share a common data source such as a database. This can be quite good an approach, allowing consistency (data entered once), integrity, scalability and real time access to changes. However:
  - Their may be some limits to how distributed the system can be, since each application needs a rapid access to the common data source
  - It may force too much integration onto the applications, which now have to agree on data structures and schemas they share. This can be problematic especially with COTS software
  - Does not provide for functional integration: if the data required is not stored but computed by another system, this solution does not help

Data integration

- Process to process communications: an application can send or receive data to or from another application in the system. This allows data sharing, wide distribution, real time update, distributed databases, independent data structure etc. This also allow “remote procedure calls”, where the called application processes the request and returns a computed result.

  However:

  - This type of integration requires an application level interface, which can be difficult to add to a program, or even impossible to add in COTS software
  - There is a need for a common protocol standard between all the applications

Data integration

In addition to the technical requirements for sharing data, there is a need for a common semantic of the shared data. Having access to some other applications data is not a goal in itself, we have in addition to ensure that the meaning of the shared data is also carried over to the other application. The same holds for functional integration, where the called procedure must perform the expected computation, that is, the semantic of the procedure must be accessible to the calling applications.

Example: markup languages

One example of a technique that has been a successful component to integrated systems are markup languages.

Markup languages date back to the 60’s : Generalized Markup Language (GML) at IBM in 1969, first Standard Generalized Markup Language (SGML) working draft in 1980. They have really reached a large audience with HTML and then XML

A web page carries its own formatting semantic and can be rendered by any HTML viewer. Coupled with the http protocol, it can be widely distributed in a seemingly platform independent way

Web services

In a nutshell, a web service is a mechanism that allow the remote invocation of methods in a language-neutral, environment-neutral way.

From a technical perspective, this is not a breakthrough. Many technologies have provided the same mechanism in the past, for example CORBA, COM, JINI etc. These other mechanisms are not in reality environment-neutral, or not language-neutral, or where not mature enough, or the domain wasn't quite ready for it yet… Web services seem to offer a good solution to a technical market that's ready for it, and are well positioned to become *the* standard.

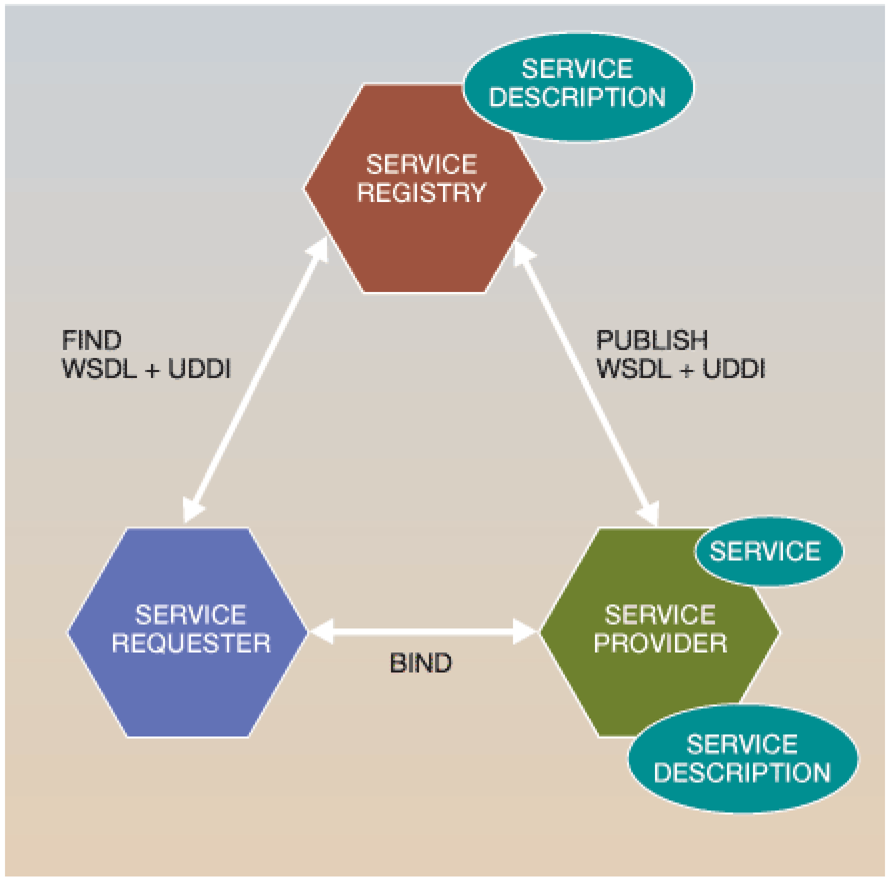
Web services

Web services require a service provider and a service requester. It also support the notion of service registry.

The service provider publishes its service on the service registry. The service requester queries the service registry to find a provider for the required service. If a match is found, the service registry sends the details to the service requester, which can then bind to the service provider

Web services

Figure 1 Web services actors, objects, and operations



http://www.research.ibm.com/journal/sj/412/gottschalk.html

Web services

In order for web service to work, many pieces must be in place: this is sometimes called the web service programming stack:

Required:

- Network and network level (communication) protocol
- invocation protocol
- Web service description

Useful:

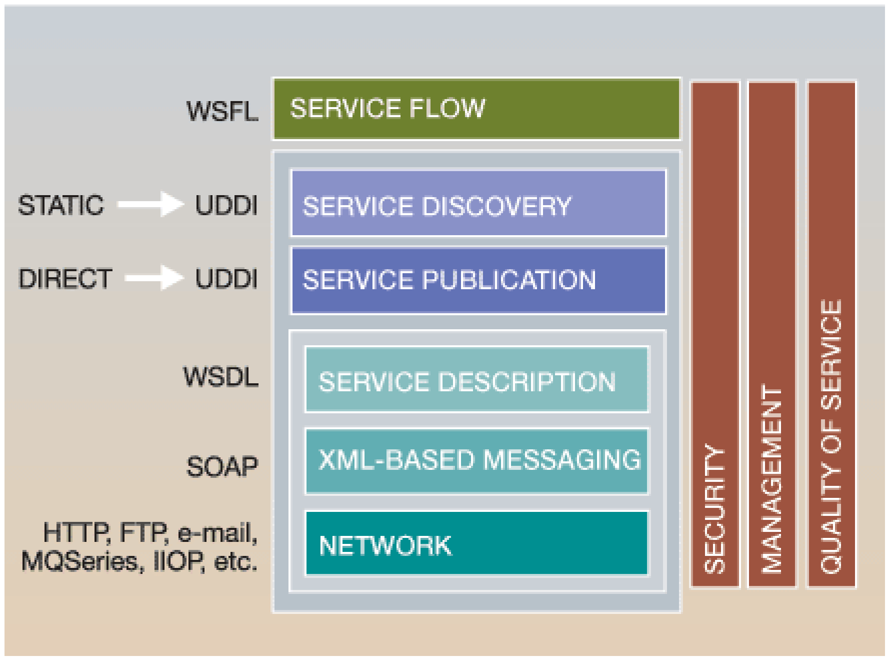
- Service publication
- Service discovery

Nice to have:

- Security (e.g. *XKMS:* XML Key Management Specification**),** Transactions (e.g. *XAML:* Transaction Authority Markup Language), Service flow (e.g. *WSFL:* Web Service Flow Language) etc.

Web services

Figure 2 Web services programming stack



http://www.research.ibm.com/journal/sj/412/gottschalk.html

Web services

The success of web services is due to the fact that many of these components are now widely available for most platforms and so if the right components are chosen, little need to be added to get a full blown web services implementation.

This leads to what is aimed at when we want “platform independence”, which is really “platform availability” for most common platform and interoperability.

Web services

The protocol level: the hyper text transfer protocol, run over TCP/IP

In theory, any network level protocol can be used, as long as it can carry data back an forth. In practice, the internet protocol is so widely available that it is the obvious choice.

Running over TCP/IP, the http protocol is also very widely deployed and gives us what we need.

Using these standards allow to get these components “for free” and to get components that are robust and widely available.

Web services

Remote invocation protocol: Simple Object Access Protocol (SOAP):

SOAP is managed by the W3C. From the draft W3C specification: “SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.”

http://www.w3.org/TR/soap

Web services

Here too, SOAP was defined separately from web services are is relatively standard and was (somewhat) widely implemented on most platform. For our purpose, it is a mechanism that will allow us to invoke any service seen as a procedure, passing any parameters of any types, and send the answer to the request. More precisely, it will allow us to define the service, its name, its parameters, its type, and it will use http to carry the request and the response back and forth.

Web services

Web Service Description Language (WSDL):

"WSDL is a specification defining how to describe web services in a common XML grammar. WSDL describes four critical pieces of data:

- Interface information describing all publicly available functions
- Data type information for all message requests and message responses
- Binding information about the transport protocol to be used
- Address information for locating the specified service"

(Web Services Essentials, O'Reilly)

Web services

With what was described so far, we have enough to have fully functional web service, as long as we know what service to call and what is their location. This is often the case in a practical system design.

Examples: http://www.xmethods.com/

As an example, here is a (piece of a) WSDL definition of a stock quote service, followed by a sample request/response, taken from

http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html?page=2

Service Advertisement

```
<?xml version="1.0"?>
<definitions name="StockQuote"
               targetNamespace="http://example.com/stockquote.wsdl"
               xmlns:tns="http://example.com/stockquote.wsdl"
               xmlns:xsd1="http://example.com/stockquote.xsd"
               xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
               xmlns="http://schemas.xmlsoap.org/wsdl/">
   <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/1999/XMLSchema">
     <element name="TradePriceRequest">
      <complexType>
       <all>
        <element name="tickerSymbol" type="string"/>
       </all>
      </complexType>
     </element>
     <element name="TradePrice">
      <complexType>
       <all>
        <element name="price" type="float"/>
       </all>
      </complexType>
     </element>
    </schema>
   </types>
```

```
<message name="GetLastTradePriceInput">
  <part name="body" element="xsd1:TradePriceRequest"/>
</message>
<message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
 <operation name="GetLastTradePrice">
  <input message="tns:GetLastTradePriceInput"/>
   <output message="tns:GetLastTradePriceOutput"/>
 </operation>
</portType>

<binding name="StockQuoteSoapBinding"
        type="tns:StockQuotePortType">
<soap:binding style="document"
 transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
   <soap:operation
    soapAction="http://example.com/GetLastTradePrice"/>
     <input>
       <soap:body use="literal"
        namespace="http://example.com/stockquote.xsd"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
     </input>
    <output>
     <soap:body use="literal"
       namespace="http://example.com/stockquote.xsd"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
   </operation>
</binding>
```

```
<service name="StockQuoteService">
  <documentation>My first service</documentation>
   <port name="StockQuotePort" binding="tns:StockQuoteBinding">
    <soap:address location="http://example.com/stockquote"/>
   </port>
 </service>

</definitions>

 <binding name="StockQuoteServiceBinding"
  type="StockQuoteServiceType">
   <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
       <operation name="getQuote">
        <soap:operation
      soapAction="http://www.getquote.com/GetQuote"/>
       <input>
        <soap:body type="InMessageRequest"
         namespace="urn:live-stock-quotes"
         encoding="http://schemas.xmlsoap.org/soap/encoding/"/>
       </input>

….
```

A SOAP enveloped request to the StockQuote service

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
    <m:GetLastTradePrice
     xmlns:m="Some-URI">
      <symbol>MOT</symbol>
    </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A SOAP enveloped response to the StockQuote service

```
HTTP/1.1 200 OK Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 <SOAP-ENV:Body>
  <m:GetLastTradePriceResponse
   xmlns:m="Some-URI">
   <Price>14.5</Price>
  </m:GetLastTradePriceResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Web services

UDDI (Universal Description, Discovery and Integration Service)

“The Universal Description, Discovery and Integration (UDDI) specifications define a registry service for Web services and for other electronic and non-electronic services. A UDDI registry service is a Web service that manages information about service providers, service implementations, and service metadata. Service providers can use UDDI to advertise the services they offer. Service consumers can use UDDI to discover services that suit their requirements and to obtain the service metadata needed to consume those services.”

www.uddi.org

Web services

UDDI (Universal Description, Discovery and Integration Service)

“The UDDI specifications define:

a) SOAP APIs that applications use to query and to publish information to a UDDI registry

b) XML Schema schemata of the registry data model and the SOAP message formats

c) WSDL definitions of the SOAP APIs

d) UDDI registry definitions (technical models - tModels) of various identifier and category systems that may be used to identify and categorize UDDI registrations”

www.uddi.org

Web services

Other characteristics

A variety of other enhancements to web services have been suggested. Some are simple proposal, some have been already implemented and are on their ways to standardization.

- Web Service Flow Language is used to specify the composition of web services in order to create a new, larger service
- XML Key Management Specification is an attempt to integrate PKI to web services
- …

Web services

Using a technology such as web service, one can usually at reasonable cost wrap a “service” around an existing application and publish this service widely for general usage. Modern languages offer all the tools needed to rapidly develop and/or use such a technology. Web servers such as Apache have built in support for the technology, and it is at the heart of Microsoft's .NET initiative.

There is an obvious large overhead in such a heavy, text based, verbose mechanism, but to many, the benefits outweigh the costs.