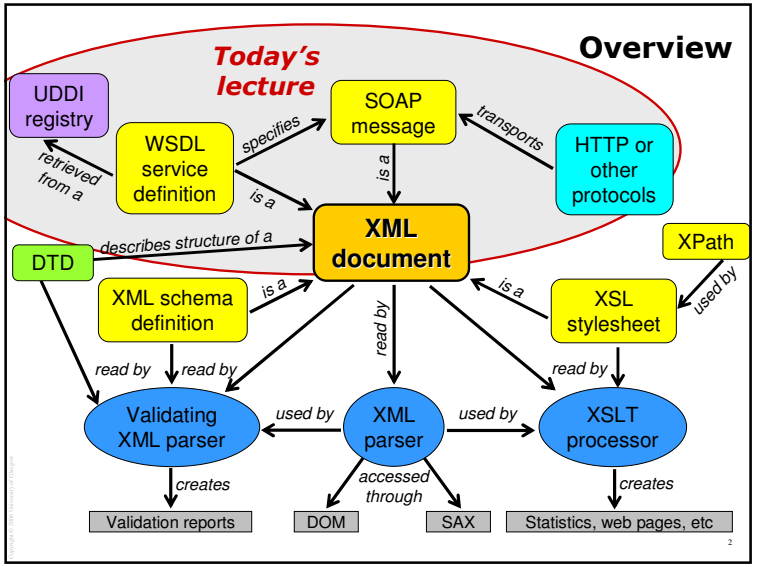
Web Services

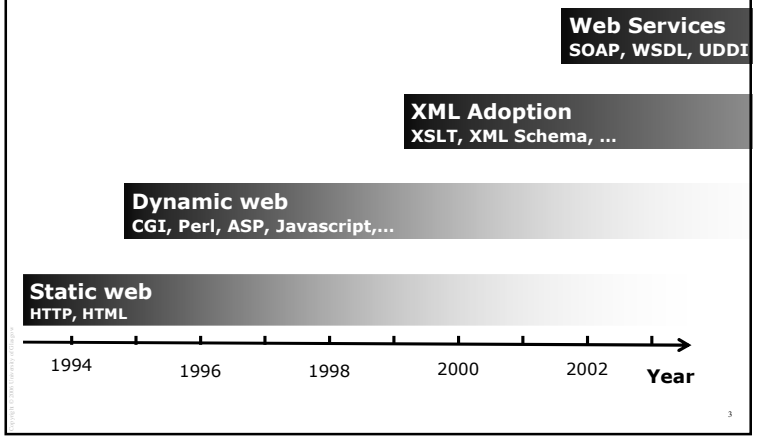
Grid Computing (M)
Lecture 6
Olufemi Komolafe (femi@dcs.gla.ac.uk)
20 January 2006

UNIVERSITY of GLASGOW

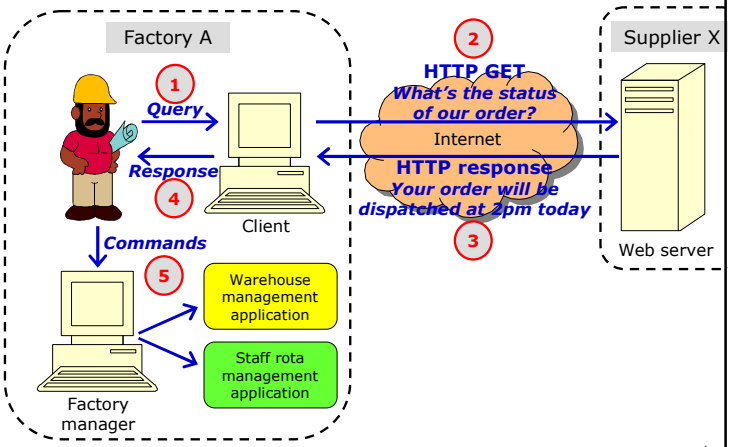
Overview



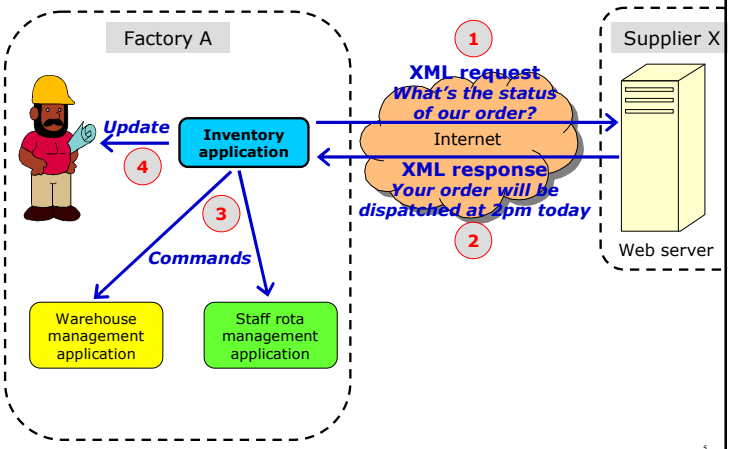
Web Technologies & Standards Evolution



The Web Today: Human Centric



Web Services: Application Centric



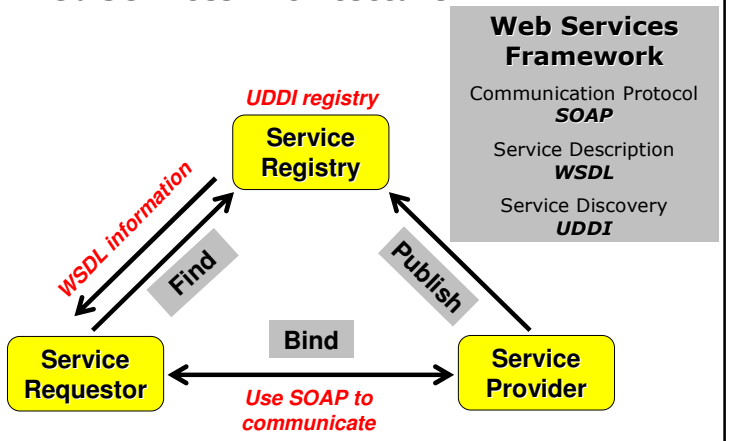
Key Steps

1. Supplier X, the ***service provider***, implements a service (to provide dispatch information for order)
2. The service provider publishes details about the service in a centralised ***service registry***
3. Factory A's inventory application, the ***service requestor***, looks up service registry to see if Supplier X provides a service to give the details of an order
   - If so, information needed to invoke the service is retrieved
4. The inventory application invokes the service & obtains the details it requires about the outstanding order

Web Services Definition

- Software service
  - Accepts requests & returns responses
    - C function, Java object etc.
  - Application ≈ well-orchestrated set of services
- Web services
  - "Web Services" ≈ "Web of Services"
  - New breed of software components that is language, platform & location independent
  - XML-based building blocks for next generation of distributed applications

Web Services Architecture



SOAP

- **Simple Object Access Protocol?**
- a communication protocol
- a format for sending messages
- designed to communicate via Internet
- platform independent
- language independent
- written in XML
- simple and extensible

SOAP Message Structure

SOAP message is ordinary XML message containing

Envelope element (mandatory)
- root element
- identifies XML document as SOAP message

Header element (optional)
- contains header information

Body element (mandatory)
- contains call & response information

Fault element (optional)
- contains information on any errors that occurred when processing the message

These elements in usual namespace for SOAP envelope

http://schemas.xmlsoap.org/soap/envelope

Typical SOAP Envelope Namespaces

SOAP envelope namespace

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

***SOAP-ENV* identifies elements & attributes from SOAP envelope namespace**
e.g. **elements** include *SOAP-ENV:Header*, *SOAP-ENV:Body* etc.
attributes include *SOAP-ENV:mustUnderstand*, *SOAP-ENV:EncodingStyle etc.*

XML Schema instance namespace

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

***xsi* identifies elements & attributes from XML schema instance namespace**
e.g. **attributes** include *xsi:type* (specifies data type of an element)

XML Schema definition namespace

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

***xsd* identifies elements & attributes from XML schema namespace**
e.g. **elements** include *xsd:string, xsd:boolean* (used as values for xsi:type attribute)

SOAP Message Structure

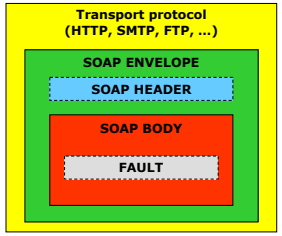
```
<?xml version="1.0" encoding = "UTF-8"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

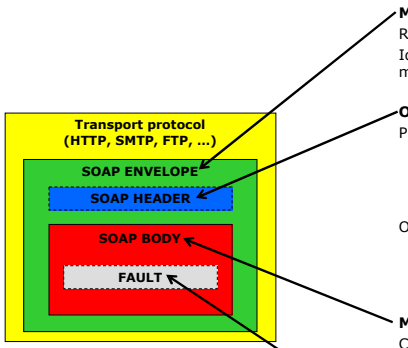
   <SOAP-ENV:Header>
      ...
   </SOAP-ENV:Header>

   <SOAP-ENV:Body>
      ...
      <SOAP-ENV:Fault>  ...  </SOAP-ENV:Fault>
      ...
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```



SOAP Message Structure



Mandatory
Root element
Identifies XML document as SOAP message

Optional
Processed by intermediate nodes
- *mustUnderstand*: compulsory to process header?
- *actor:* target Header to particular endpoint

Orthogonal to payload
- Info for authentication, transactions, routing, payment etc.

Mandatory
Contains payload of SOAP message

Optional
Holds error & status information

SOAP over HTTP

2 HTTP request methods
- GET
  - Parameters sent in URL
  - Typically used to request web page from web server
- POST
  - Send data to server in payload that come after HTTP header
  - Payload can be of indefinite length
  - Used by SOAP

POST /SOAPendpoint HTTP/1.1
Host: 130.29.25.3
Content-Type: text/xml *Content is XML*
Content-Length: nnn *Typically large… XML bloat :-(*
SOAPAction: intentURI

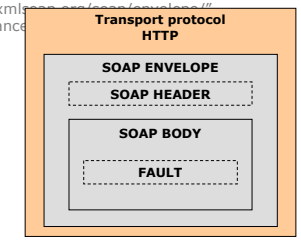
HTTP method to be initially executed; allows filtering of SOAP request messages in HTTP

SOAP over HTTP is basically "HTTP post with XML payload"

```
<?xml version="1.0" encoding = "UTF-8"?>

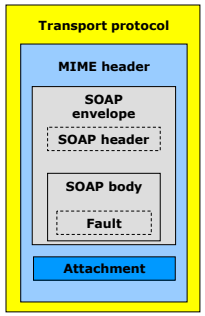
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
    <SOAP-ENV:Header>
        …
    </SOAP-ENV:Header>

    <SOAP-ENV:Body>
        …
        <SOAP-ENV:Fault>  …  </SOAP-ENV:Fault>
        …
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

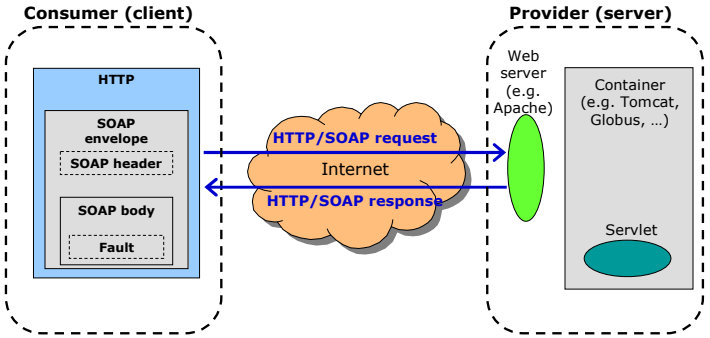


SOAP Attachments

- Large quantities or binary data may not fit well into an XML SOAP message
  - Send "out of band" by attaching to SOAP message
    - c.f. email attachments



Using SOAP



Servlet is server side application running in container that processes SOAP information

Web server extracts information from HTTP/SOAP envelope and directs it to appropriate service being hosted in container

SOAP Request

```
<?xml version='1.0' encoding = 'UTF-8'?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

   <SOAP-ENV:Body>
      <ns:methodname xmlns:ns="URI_methodname">
         <parameter1>value of parameter1</parameter1>
         <parameter2>value of parameter1</parameter2>
         ...
      </ns:methodname>
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Namespace for method

Method parameter(s) appear as sub-element(s)

***Body** contains single element which corresponds to remote method name*

methodname(parameter1, parameter2, ...)

Example SOAP Request

```
<?xml version="1.0" encoding = "UTF-8"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

      <SOAP-ENV:Body>
         <dis:getdispatchinfo xmlns:dis="URI_getdispatchinfo">
            <ordernumber xsi:type="xsd:string">xxxx</ordernumber>
         </dis:getdispatchinfo>
      </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

getdispatchinfo(ordernumber)

ordernumber assigned to XML Schema xsd:string data type

value of ordernumber

Example SOAP Response

```
<?xml version="1.0" encoding = "UTF-8"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

      <SOAP-ENV:Body>
         <dis:getdispatchinfoResponse xmlns:dis="URI_getdispatchinfo">
            <dispatchday xsi:type="xsd:string">yyyy</dispatchday>
         </dis:getdispatchinfoResponse>
      </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

*conventionally methodname**Response***

SOAP Fault Handling

In event of error (e.g. client requests method that doesn't exist at server), *Body* element will contain *Fault* element

```
<?xml version="1.0" encoding = "UTF-8"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
      <SOAP-ENV:Body>
         <SOAP-ENV:Fault>
            <faultCode>....</faultCode>
            <faultString>....</faultString>
            <faultActor>....</faultActor>
            <detail>....</detail>
         </SOAP-ENV:Fault>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Text code to indicate class of error

Human-readable error explanation

Text string indicating fault causer

To carry application-specific error messages

SOAP Implementations

Several implementations of SOAP specifications

- Microsoft with Visual Basic, C#, .NET languages/environments
- Apache Axis with Java
- SOAP::Lite with Perl
- Systinet WASP with C++, Java
- GLUE with Java

See appropriate website for further information

More SOAP information

http://www.w3schools.com/soap/default.asp

WSDL

- **Web Services Description Language**
- Describes 4 critical pieces of information
  - interface information describing publicly available functions
  - data type information for all message requests & responses
  - binding information about the transport protocol to be used
  - address information for locating the specified service
- Represents contract between service requestor & service provider
- Verbose
  - usually auto-generated

WSDL Document Structure

<definitions>	*Root WSDL element. Defines web service name, declares namespaces etc.*
<types> … </types>	*Defines all data types used by web service*
<message> ... </message>	*Describes messages to be exchanged*
<portType> … </portType>	*Defines a web service, the operations that can be performed & the messages involved.*
<binding> …</binding>	*Defines the message format and protocol details for each port*
<service> …</service>	*Defines address for invoking specified service*
</definitions>	

WSDL Document Structure

```
<definitions>
    <types>
        <schema></schema>*
    </types>
    <message>*
        <part></part>*
    </message>
    <portType>*
        <operation>*
            <input></input>
            <output></output>
        </operation>
    </portType>
    <binding> *
        <operation>*
            <input></input>
            <output></output>
        </operation>
    </binding>
    <service> *
        <port></port>*
    </service>
</definitions>
```

*- multiple occurrences possible

<definitions>

- Mandatory root element of all WSDL documents
- Defines name of the web service
- Declares multiple namespaces used in document

<types>

- Describes all data types exchanged between the client and server
  - i.e. the data types passed to/from the web service
- These defined types are used by ***message*** elements
- Comprises of ***schema*** element(s)
  - must follow rules for XML Schema documents
    - e.g. new elements defined using *simpleType*, *complexType* etc.
- If only simple XML Schema built-in types (e.g. strings, integers) used, ***types*** element unnecessary

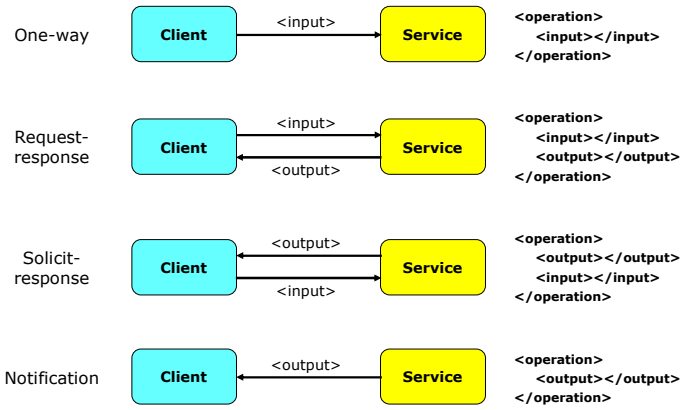
<message>

- Collection of ***message*** elements defines all messages that will be exchanged between the client & service endpoints
- A message is simply a piece of information moving between client and service
- Typical round trip method call modelled as 2 messages
  - One message for request, other for response
- Each message comprises of ***part*** element(s)
  - uses data types defined in the ***type*** element

<portType>

- Combines multiple ***message*** elements to form complete one-way or round-trip operation
- Message attribute for each input/output must correspond to name of message defined earlier in the WSDL document
- portType ≈ class, operation ≈ method in class
- Number of different operation patterns supported
  - Synchronous
    - Request-response
    - Solicit-response
  - Asynchronous
    - One-way
    - Notification

WSDL Operation Patterns



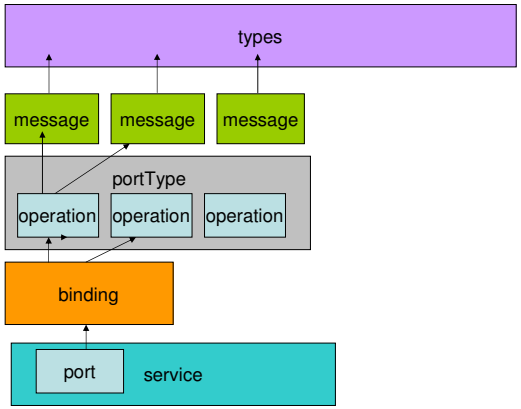
<binding>

- Describes specifics of how the service will be implemented on the wire
- Corresponds to a ***portType*** implemented using a particular protocol, typically SOAP
- WSDL includes built-in extensions for defining SOAP services
  - SOAP-specific information described in ***binding*** element
  - SOAP binding extension in WSDL mean SOAP-specific grammar can be sued in ***binding*** element

<service>

- Defines address for invoking the specified service
  - Typically URL for invoking SOAP service
- Web service is grouping of one or more ***port*** elements
  - ***port*** element has name attribute that provides unique identifier for endpoint
  - Content of ***port*** element dependent on mechanism, i.e SOAP, HTTP etc.

WSDL Overview



Basic WSDL Example 1/2

Web Service has single publicly available function, sayHello(). Input parameter is a string & output is a greeting.
e.g. sayHello(world) returns "Hello, world!"

WSDL namespace set as default namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
   targetNamespace="http://www.foo.org/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.foo.org/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Namespace for SOAP elements in WSDL doc

Request message

Request message name

```
   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>
```

Specifying function parameter

Respond message

```
   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>
```

Specifying return value

Defines single operation called *sayHello*, single input & output message

```
   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>
```

Input & output messages (defined above)

```
   <binding name="Hello_Binding" type="tns:Hello_PortType">
      <soap:binding style="rpc"
         CONTINUED ON NEXT SLIDE....
```

Example from "Web Services Essentials" by E. Cerami (O'Reilly, 2002)

Basic WSDL Example 2/2

Refers to portType defined above

```
<binding name="Hello_Binding" type="tns:Hello_PortType">
   <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </input>
      <output>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </output>
   </operation>
</binding>
```

RPC-styled SOAP message

Indicates HTTP used to transport SOAP

Specifies soapAction HTTP header

Specifies SOAP encoding style & URN associated with specified service

binding supplies information on how *sayHello* operation will be transported over the Internet

Specifies location of service

```
<service name="Hello_Service">
   <documentation>WSDL File for HelloService</documentation>
   <port binding="tns:Hello_Binding" name="Hello_Port">
      <soap:address
         location="http://localhost:8080/soap/servlet/rpcrouter"/>
   </port>
</service>
</definitions>
```

Specifies local host address for the Apache SOAP rpcrouter servlet

Overview of HelloService.wsdl

<definitions>:
The HelloService

<message>:
1. sayHelloRequest:firstName parameter
2. sayHelloResponse:greeting return value

<portType>:
sayHello operation that consists of a request/response service

<portType>:
Directions to use SOAP & HTTP

<portType>:
Service available at
http://localhost:8080/soap/servlet/rpcrouter

Example of Corresponding SOAP Message?

UDDI

- **Universal Description, Discovery & Integration**
- Like the "phone directory" of web services
- Technical specification for building distributed directory of business and web services
  - Data stored in specific XML format
  - UDDI specification includes API details for searching existing data and publishing new data
- 3 parts to technical architecture
  - UDDI data model
    - XML Schema for describing businesses & web services
  - UDDI API
    - API for searching & publishing UDDI data
  - UDDI cloud services
    - Operator sites that provide implementations of the UDDI specification and synchronise all data regularly

UDDI Registries

- Three registries
  IBM – http://www-3.ibm.com/services/uddi/
  Microsoft – http://uddi.microsoft.com/
  SAP - http://uddi.sap.com/
- IBM and Microsoft both include a live registry and a test registry
- Regularly synchronised
  - if you publish to one, pretty soon it will be on all three

Overview

