Introduction to XML and Web Services

Open Web Standards for Interoperability

eXtensible Markup Language (XML)

- XML is not a replacement for HTML:
  - HTML was designed to format and display data:

```
<H1> INB373 Web Application Development</H1>
<em>Unit Coordinator:</em> Wayne Kelly
<p>
Prerequisite: INB271
```

  - XML was designed to give structure and meaning to data:

```
<unit>
    <code>INB373</code>
    <name>Web Application Development</name>
    <coordinator>Wayne Kelly</coordinator>
    <prerequiste>INB271</prerequisite>
</unit>
```

  - XML is a metalanguage for designing markup languages

What is XML used for?

- XML was designed to be a generic data format for sharing data between applications (often over the Internet).
- XML is:
  - extensible
  - general purpose.
  - self describing.
  - validatable.
  - human readable.
  - easy to parse and generate.
  - platform independent.
  - programming language independent.
  - object model independent.

Other Differences from HTML

- XML is extensible:
  - there are no predefined tags, you make up your own.
- All start tags must have a matching end tag.

  (and tags must be properly nested)

   This is a paragraph

  Incorrect: This text is bold and italic

   Correct: This text is bold and italic
- Each document must have a unique first element, the root node (a.k.a **document element**).
- All attribute values must be enclosed in quotation marks.

  <CITY **ZIP="01085">**Westfield</CITY>
- XML is case sensitive.

  <Message>This is incorrect</message>

  <message>This is correct</message>

XML Elements and Attributes

- Elements:
  - An XML element is made up of a start tag, an end tag, and data in between.
  - The start and end tags describe the data within the tags, which is considered the value of the element.
  - For example, the following XML element is a <director> element with the value "Matthew Dunn."

```
<director>Matthew Dunn</director>
```

- Attributes:
  - An attribute is a named simple-type definition
  - An attribute is a name-value pair separated by an equal sign (=).

```
<CITY ZIP="01085">Westfield</CITY>
```

- Differences between Elements and Attributes:
  - An element can optionally contain one or more attributes.
  - An element can optionally contain one or more other elements.
  - An attribute cannot contain other elements or attributes

XML Namespaces

- An XML namespace is a collection of names that can be used as element or attribute names in an XML document.
  - a namespace qualifies element names uniquely on the Web in order to avoid conflicts between elements with the same name.
  - a namespace is identified by a Uniform Resource Identifier (URI)
- Explicit namespace declarations (a namespace that is associated with a prefix):

```
<BOOKS>
    <bk:BOOK xmlns:bk="urn:BookLovers.org:BookInfo"
             xmlns:money="urn:Finance:Money">
        <bk:TITLE>A Suitable Boy</bk:TITLE>
        <bk:PRICE money:currency="US Dollar">22.95</bk:PRICE>
    </bk:BOOK>
</BOOKS>
```

- Default namespace declarations (a namespace that is associated with no prefix):

```
<BOOKS>
    <BOOK xmlns="urn:BookLovers.org:BookInfo">
        <TITLE>A Suitable Boy</TITLE>
        <PRICE currency="US Dollar">22.95</PRICE>
    </BOOK>
</BOOKS>
```

XML Schema

- An XML Schema is an XML-based syntax for defining the structure, content and semantics of XML documents.
- XML Document:

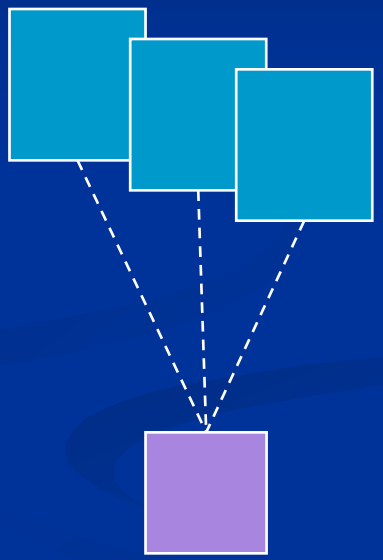
XMLFile1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<class xmlns="http://qut.edu.au/class">
  <student studentID="13429">
    <name>James Smith</name>
    <GPA>3.8</GPA>
  </student>
</class>
```

- Corresponding XML Schema (classSchema.xsd):

classSchema.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
           targetNamespace="http://qut.edu.au/class"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="class">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="student">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string" />
              <xs:element name="GPA" type="xs:decimal" />
            </xs:sequence>
            <xs:attribute name="studentID" type="xs:unsignedShort" use="required" />
          </xs:complexType>
```



XML References

Tutorials:

http://www.w3schools.com/xml/

http://www.javaworld.com/javaworld/jw-04-1999/jw-04-xml.html

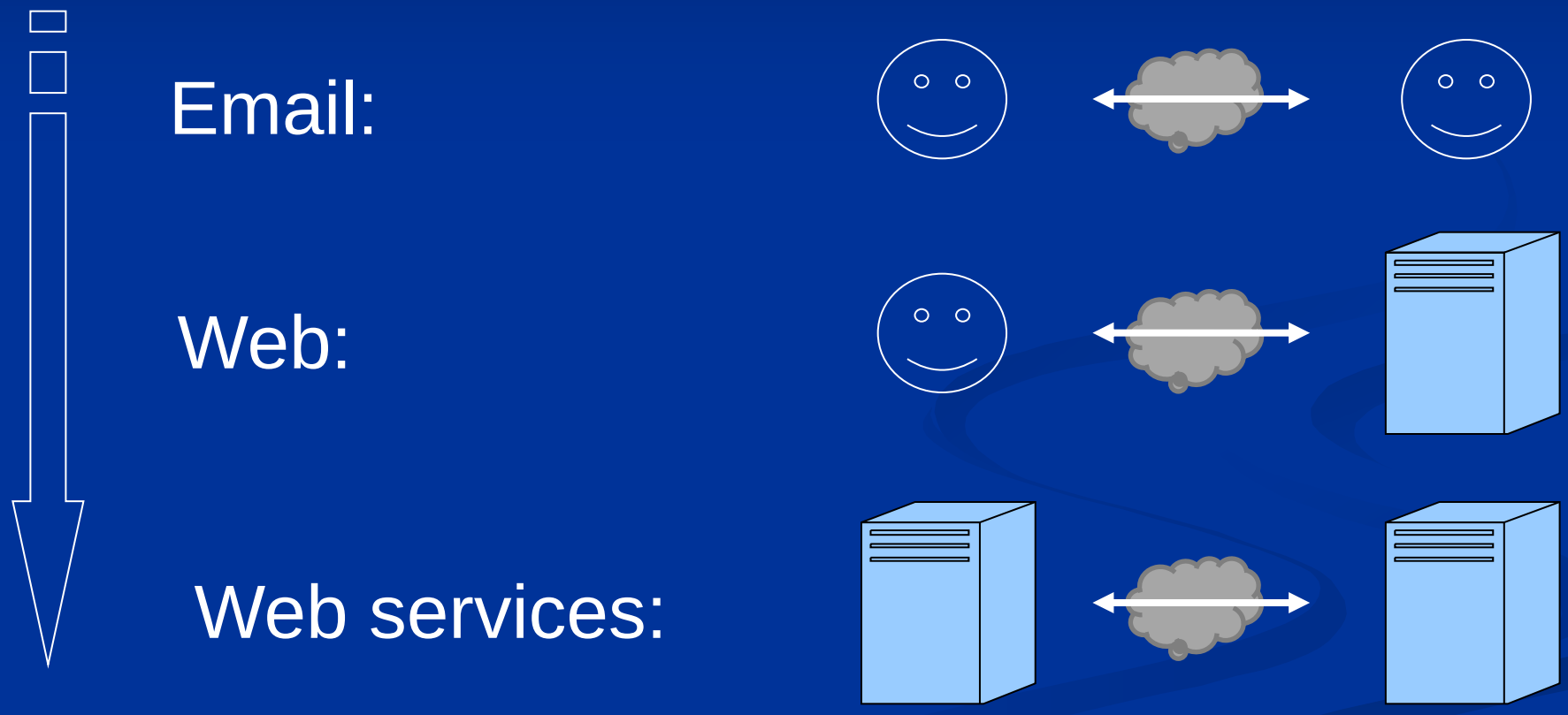
http://www.xml.com/pub/a/98/10/guide0.html

Specification:

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

Web Services

Internet's Evolution



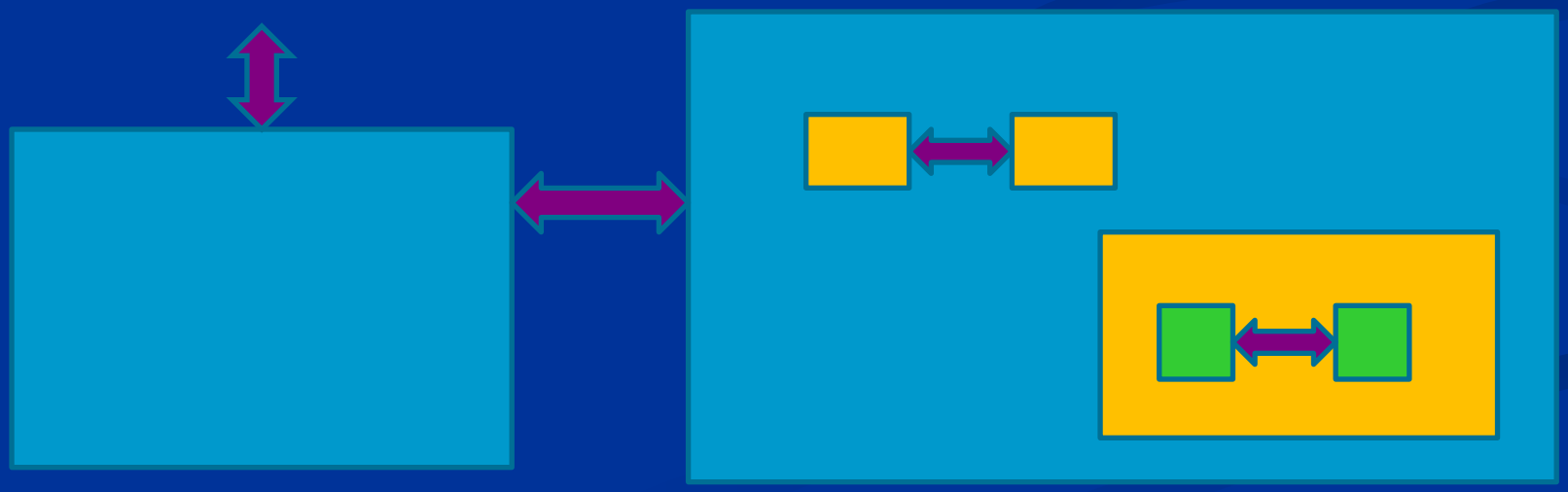
Web Service Definition

1. Designed for use by machines rather than humans
   - Expose a programmatic interface rather than a graphical user interface.
2. Use open standards to communicate in heterogeneous environments
   - Independent of vendors, programming languages, object models and operating systems.

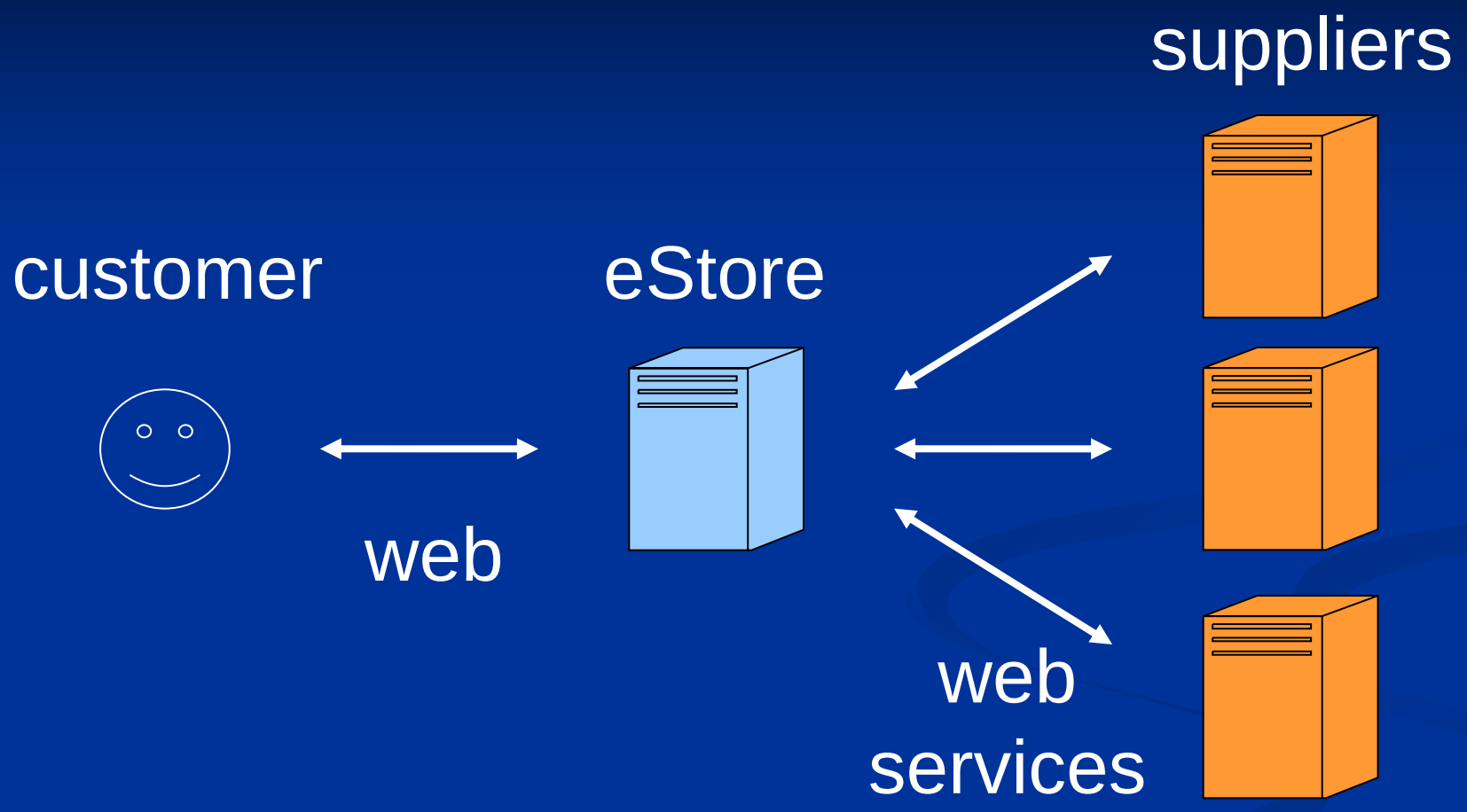
W3C definition: A "**web service**" is "a software system designed to support interoperable machine-to-machine interaction over a network".

Uses for Web Service

- Business to Business (B2B) Integration
- Business to Customer (B2C)
- Enterprise Application Integration (EAI)
- Application inter-tier communication



B2B Example: Supply Chain Automation



Why Not CORBA, DCOM, Java RMI etc.

CORBA - OMG's Common Object Request Broker Architecture
DCOM - Microsoft's Distributed Component Object Model
Java RMI - SUN's Java/Remote Method Invocation

- Too complex
- Poor interoperability
- Designed for local area networks rather than Internet
- Not standards based
- Note there have been previous attempts e.g. EDI
  - Problem fixed, not extensible
- Good introduction and motivation to WS
  http://msdn2.microsoft.com/en-au/magazine/cc188933.aspx

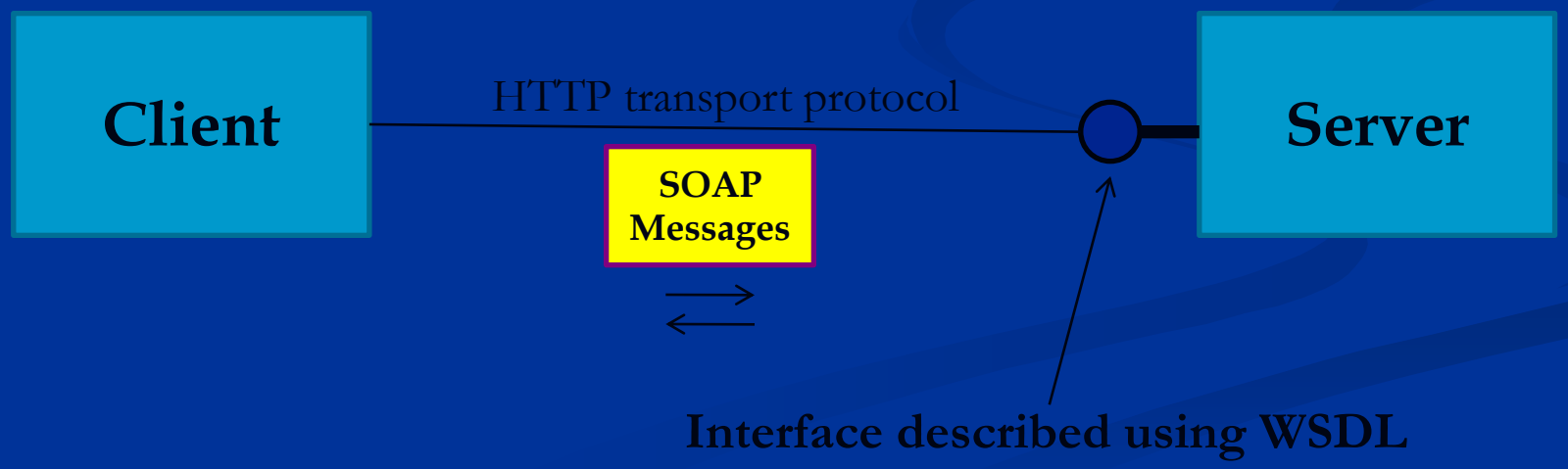
The Two Worlds of Web Services

- SOAP Web Services
  - Traditional
  - Heavier weight
  - Lots of standards
- RESTful Web Services
  - Fresh
  - Lighter weight
  - More informal usage

SOAP Web Services

SOAP Web Services

- **SOAP** (Simple Object Access Protocol)
  - an XML based protocol for web service messages.
- **WSDL** (Web Service Description Language)
  - An XML based language for describing web services
- Standards maintained by W3C



SOAP

- SOAP messages are XML
  - XML info set, not necessarily serialized as XML 1.0 document.
- SOAP specification includes rules:
  - message formatting (XML)
  - message processing
- SOAP is designed to be *Extensible*
  - To support future web service standards and protocols
  - To allow application interfaces to change/evolve.
- Independent of transport protocol (e.g. HTTP)

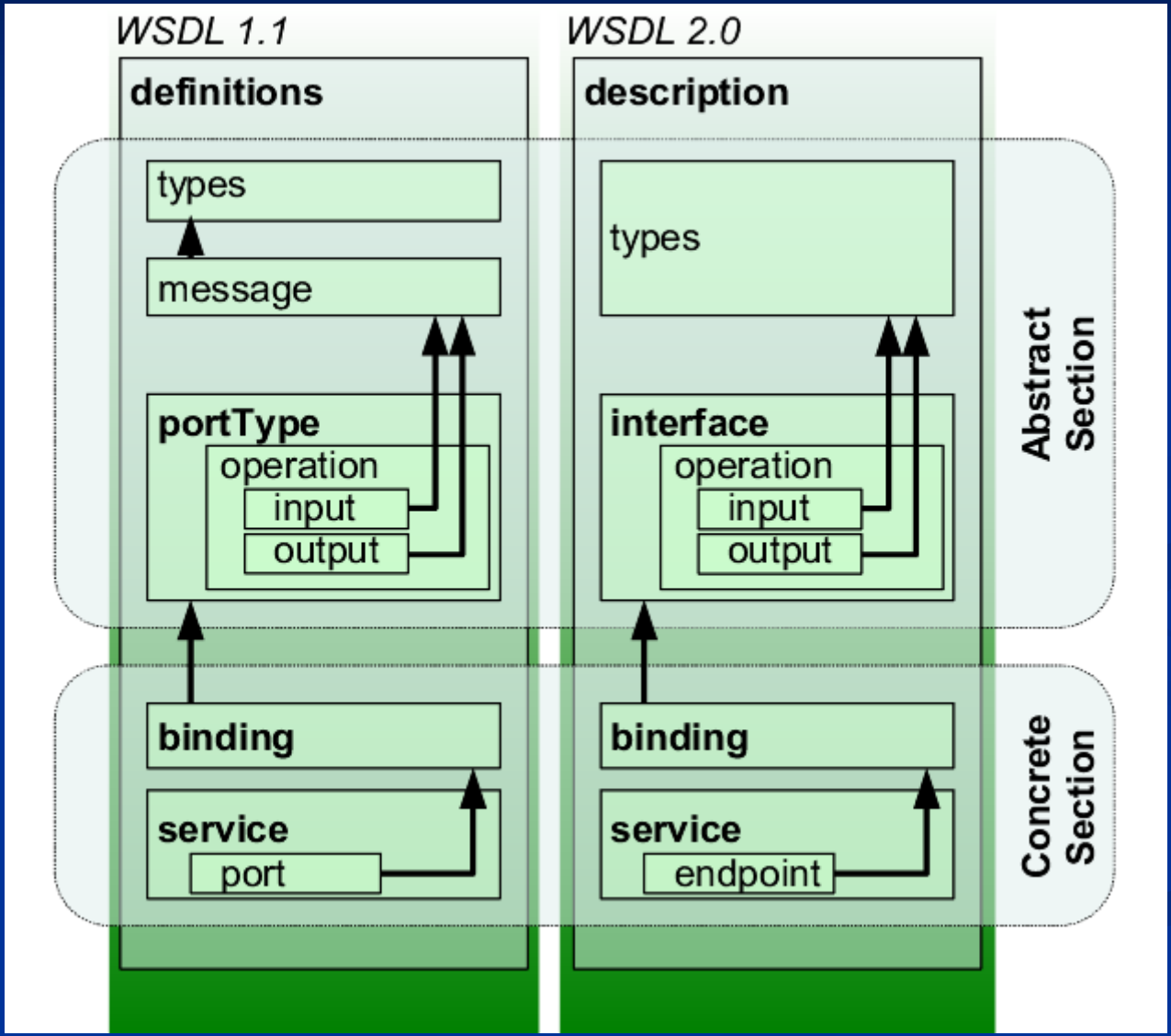
SOAP Message Format

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    Application specific content
  </env:Header>
  <env:Body>
    Application specific content
  </env:Body>
</env:Envelope>
```

WSDL

- A WSDL specification describes a web service.
- A WSDL specification is a XML document
- WSDL is designed to be *Extensible*
  - Independent of transport protocol (e.g. HTTP)
  - Independent of message format (e.g. SOAP)
  - Independent of type definition language (eg XML Schemas)

Structure of a WSDL specification



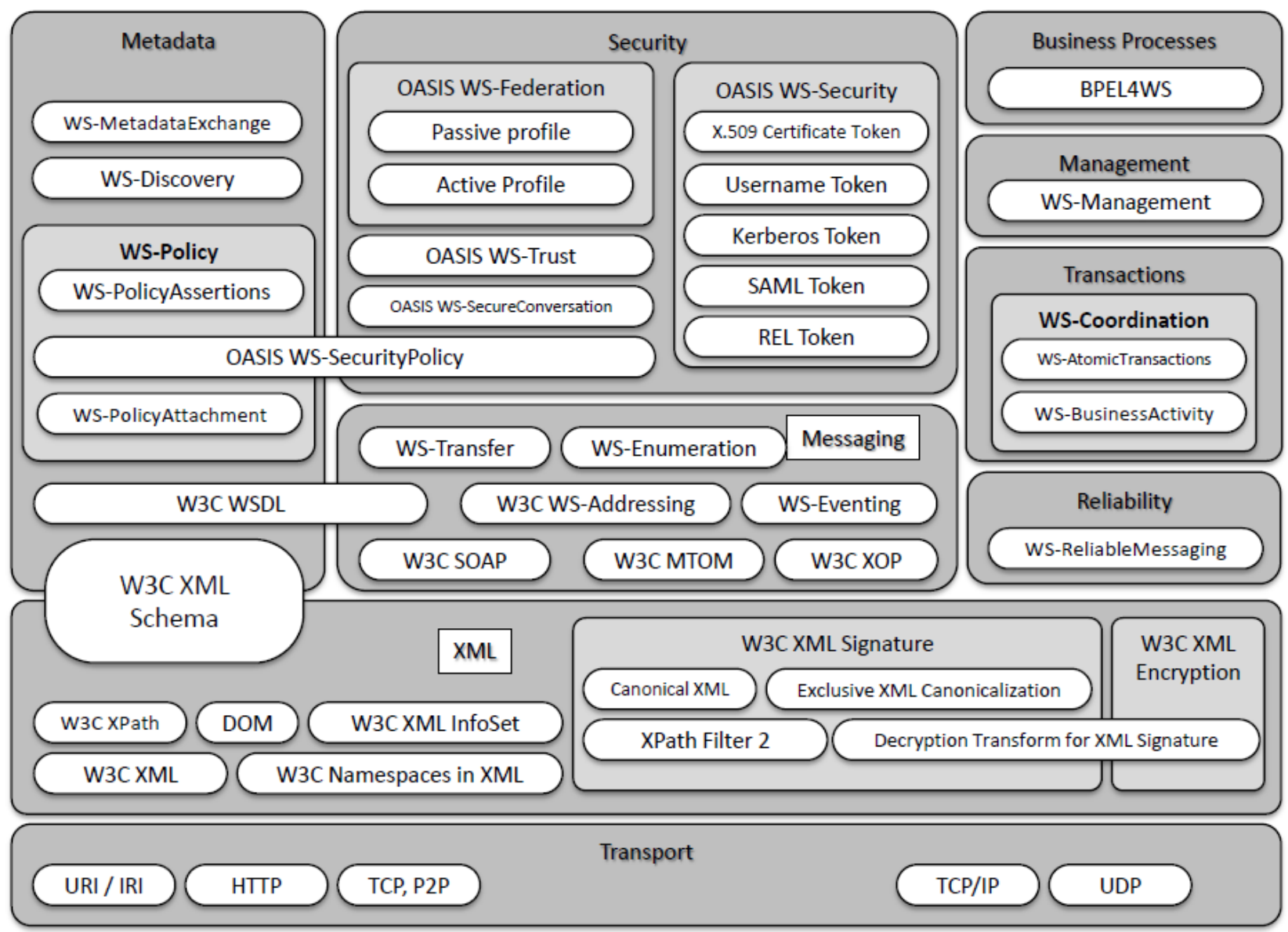
WSDL 2.0 Concrete Section

- WSDL enables one to separate the description of the abstract functionality offered by a service from concrete details of a service description such as "how" and "where" that functionality is offered.
- **Binding**
  - a *binding* specifies transport and wire format details for one or more interfaces (e.g. HTTP)
- **Endpoint**
  - An *endpoint* associates a network address with a binding (e.g. URL)
- **Service**
  - a *service* groups together endpoints that implement a common interface
- www.w3.org/TR/**wsdl**20/

WSDL 2.0 Document Format

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl">
  <types>
    type descriptions (defined using e.g. XML schema)
  </types>
  <interface name = "???" >
    list of operations with input and output types
  </interface>
  <binding name="???" interface="???" type="e.g. SOAP" protocol=" e.g. HTTP">
    binding descriptions for each operation in the interface
  </binding>
  <service name="???" interface="???">
    list of endpoints consisting of address and binding
  </service>
</description>
```

SOAP Web Service Standards



RESTful Web Services

REST (Representational State Transfer)

- Distributed software architecture invented by Roy Fielding in PhD thesis 2000.
  - Architectural Styles and the Design of Network-based Software Architectures
- *Servers*
  - host Resources
- *Resources*:
  - can be uniquely identified
  - have an internal state
- *Clients* manipulate Resources by sending and receiving messages that:
  - identify the resource(s)
  - specify one of a fixed number of *operations* to perform (e.g. create, read, update, delete)
  - may include a *representation* of the (partial) state or intended state of the resource

RESTful Web Services

- RESTful Web Services result from applying REST principles to HTTP:
- Resources are identified via URLs
- Operations are restricted to HTTP methods:
  - **`POST`** (Create)
  - **`GET`** (Read)
  - **`PUT`** (Update)
  - **`DELETE`** (Delete)
- Representation can be whatever is most convenient:
  - E.g.: JSON, XML, atom, binary
  - Leverage standard HTTP headers (e.g. content-type).

REST Request Examples

GET http://api.twitter.com/1/statuses/show/*114651506.json*

- returns the status of author with id: *114651506* in JSON format

DELETE http://api.twitter.com/1/twitterapi/*team*/subcribers.xml

- unsubscribes the authenticated user from list with id: *team*.

POST http://api.twitter.com/1/twitterapidocs/lists.xml?name=*doctors*

- creates a new list with name: *doctors* for the authenticated user.

PUT http://api.twitter.com/1/twitterapidocs/lists/doctors.xml?name=surgeons

- changes the name of the list named *doctors* to *surgeons*

http://dev.twitter.com/console

Resource Representation

- **XML**

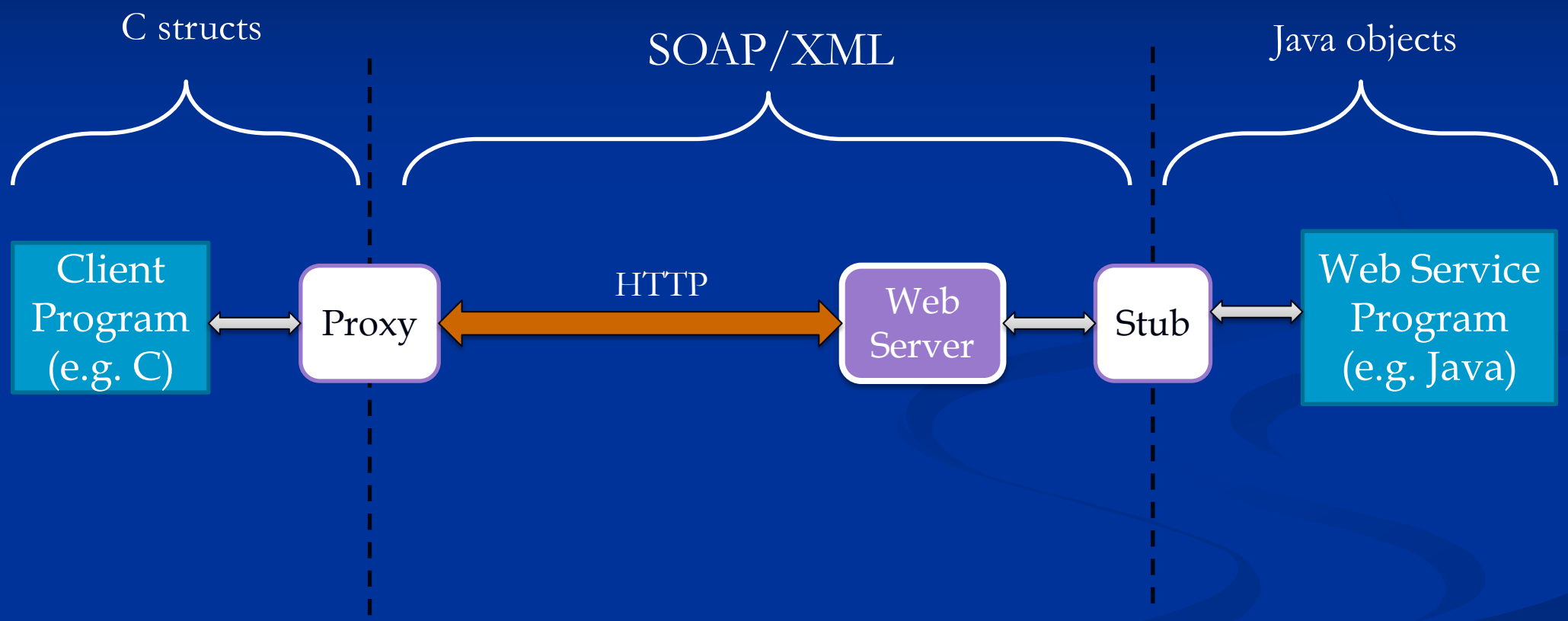
```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumber type="home">212 555-1234</phoneNumber>
  <phoneNumber type="fax">646 555-4567</phoneNumber>
</person>
```

- **JSON** (JavaScript Object Notation)

```
{
 "firstName":"John", "lastName":"Smith", "age":25,
 "address":
    {"streetAddress":"21 2nd Street", "city":"New York", "state":"NY", "postalCode":"10021"},
 "phoneNumber":
    [{"type":"home", "number":"212 555-1234"},{"type":"fax", "number":"646 555-4567"}]
}
```

Implementing and Using SOAP Web Services

Converting XML to Objects



WSDL First ?

WSDL as an implementation artefact:

1. Implement web service program
2. Auto-generate Stub and WSDL
3. Auto-generate client Proxy from WSDL
4. Implement web service client program

or, **WSDL first**:

1. Design WSDL specification
2. Auto-generate server stub from WSDL
3. Implement web service program
4. Auto-generate client Proxy from WSDL
5. Implement web service client program

WSDL First (in Practice)

1. Create abstract interface of web service program
2. Auto-generate WSDL
3. Manually refine/redesign the WSDL
4. Regenerate the Stubs from the new WSDL
5. Implement web service program
6. Auto-generate client Proxy from WSDL
7. Implement web service client program

Web Service Programming Frameworks

- .NET:
  - ASP.NET web services
  - Microsoft Windows Communication Foundation (WCF)
- Java:
  - JAX-WS
  - …
- …

Some Web Services to try ...

- Terraserver http://terraserver-usa.com/
- MS Map Point:

  http://www.microsoft.com/mappoint/products/webservice/default.mspx
- Google http://www.google.com/apis/
- Flickr www.flickr.com/services/api/
- Amazon http://aws.amazon.com

Everyone's Talking Web Services

- W3C : http://www.w3.org/2002/ws/
- Microsoft: http://msdn.microsoft.com/webservices/
- Sun: http://java.sun.com/webservices/
- IBM: http://www-106.ibm.com/developerworks/webservices/
- Borland: http://www.borland.com/webservices/

Web Service Issues

- Specification evolution – is the end in sight?
- Security
- Twist: many organisations using web services for loosely coupling intranet systems
- Killer application?