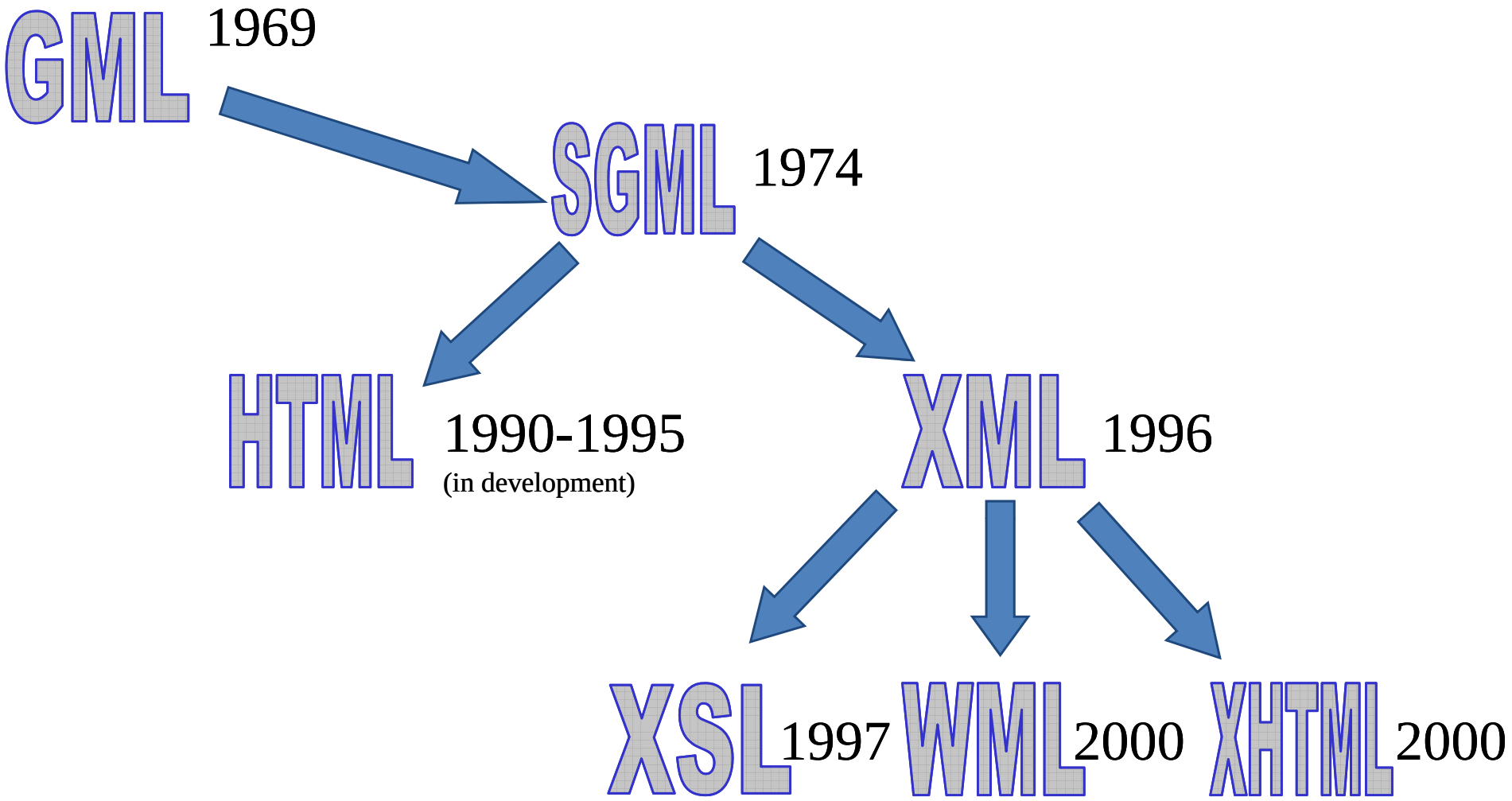
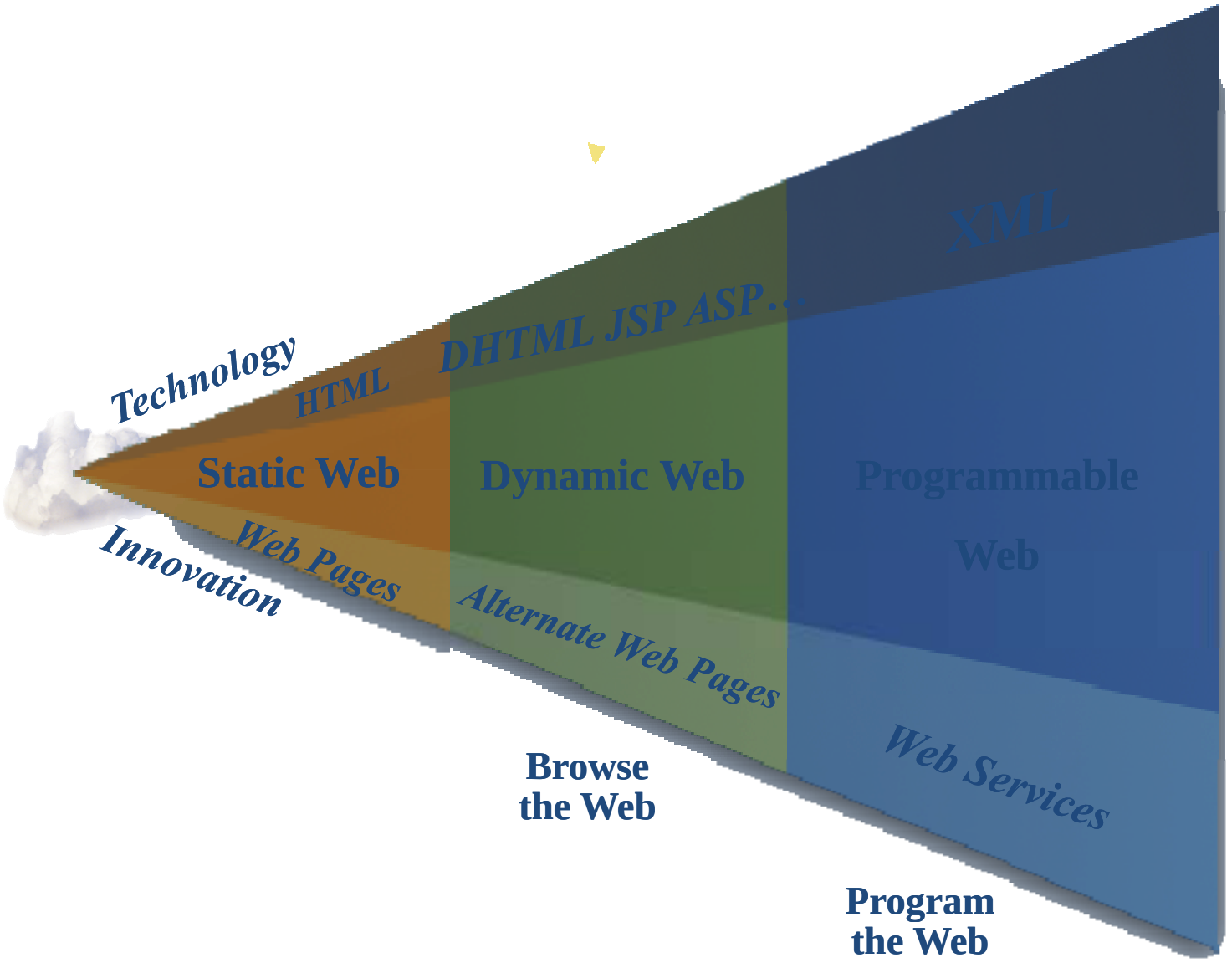
CSP2103-4102: Markup Languages

Lecture 10: Web Services

Introduction

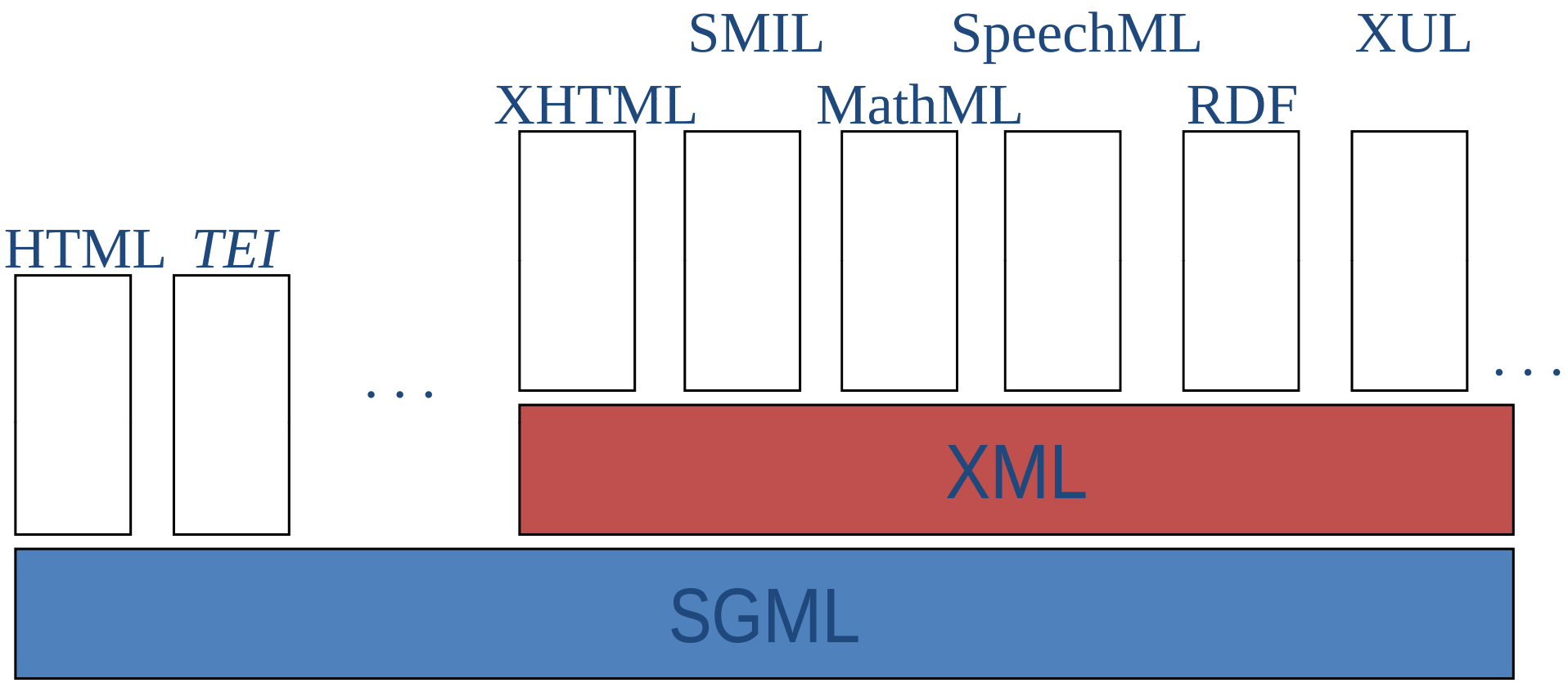


Web Evolution



(Shuying Wang, 2004)

The XML Family Tree



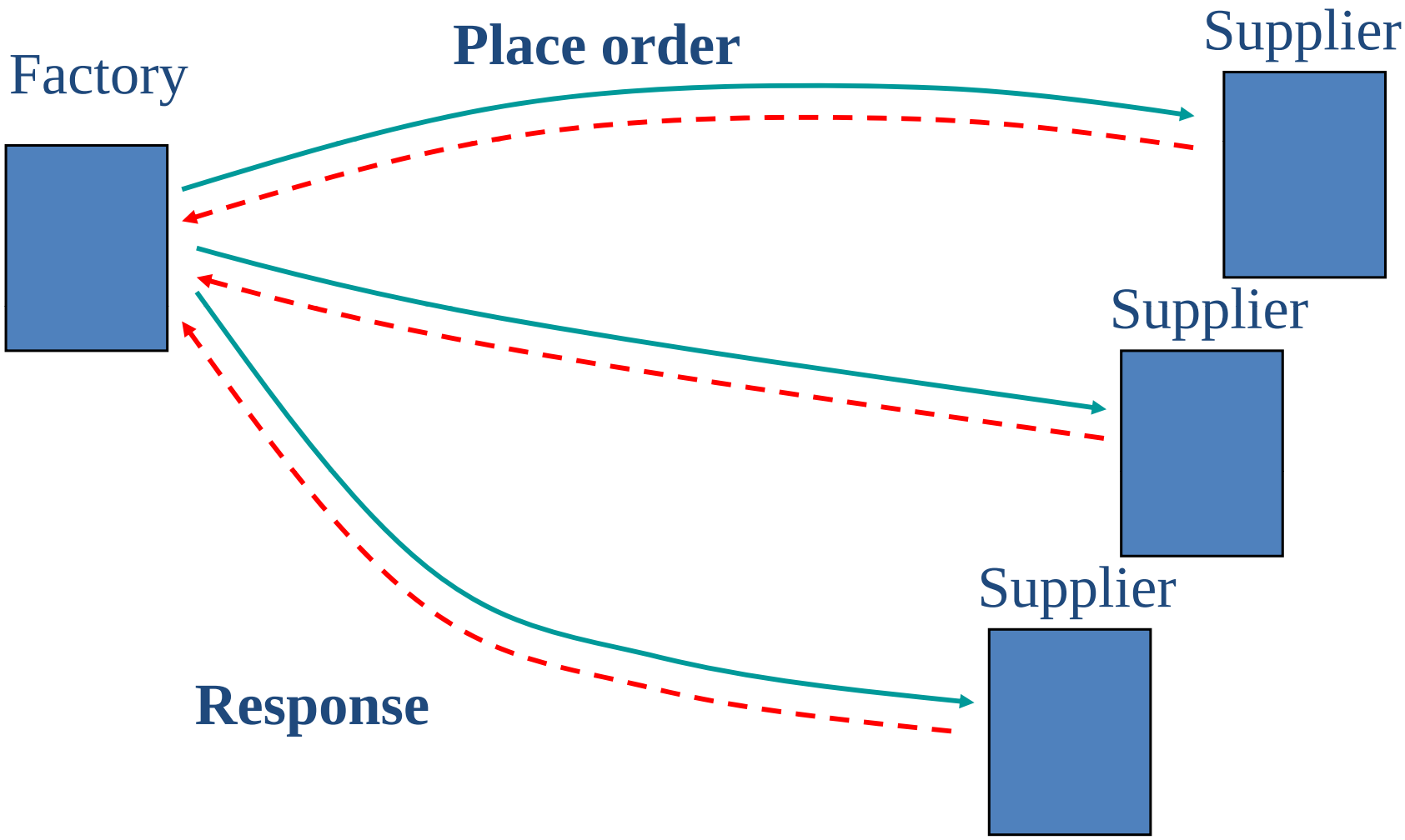
XML Summary

- An integration tool for mixing different types of data
- A universal format for exchanging data between machines
- A framework for distributing information on the Web
- Allowing developers to create their own markup languages (using xml and xslt)

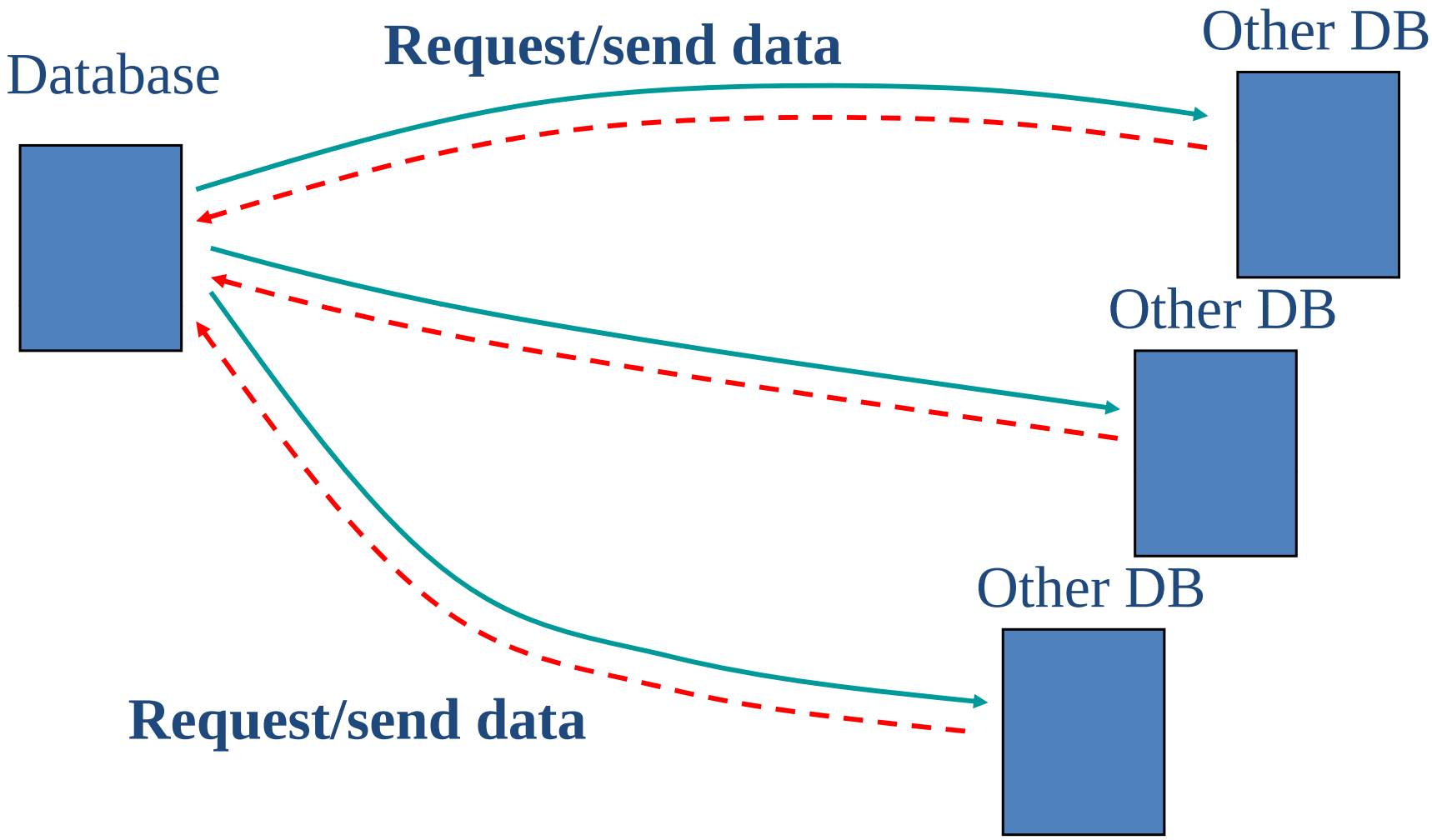
XML Uses: Data Abstraction

- XML as a universal format for data interchange
  - Machines exchange data as XML-format messages
  - Eliminates proprietary data formats
  - Lots of XML processing software available

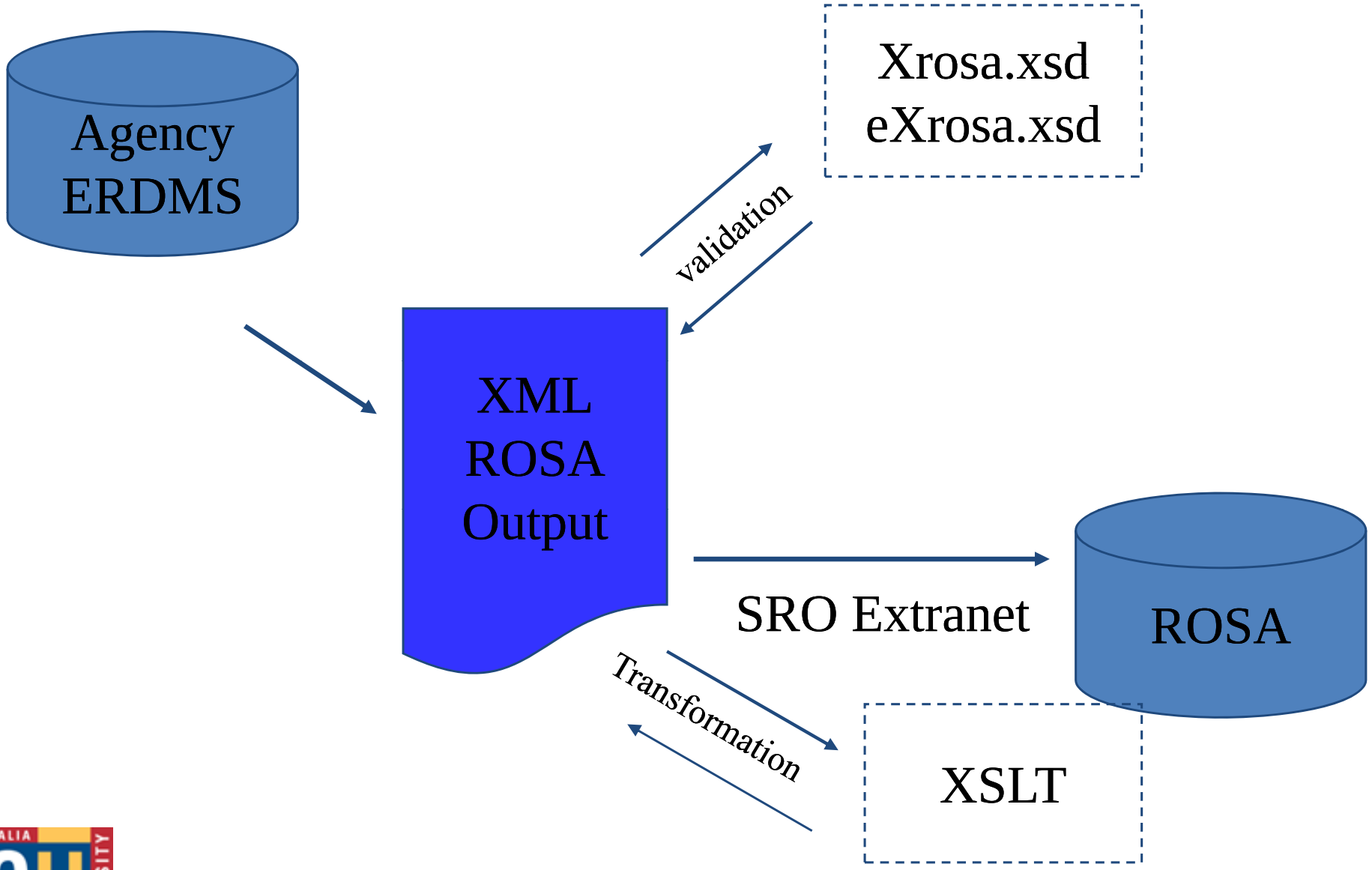
XML Messaging: Business



XML Messaging: Database



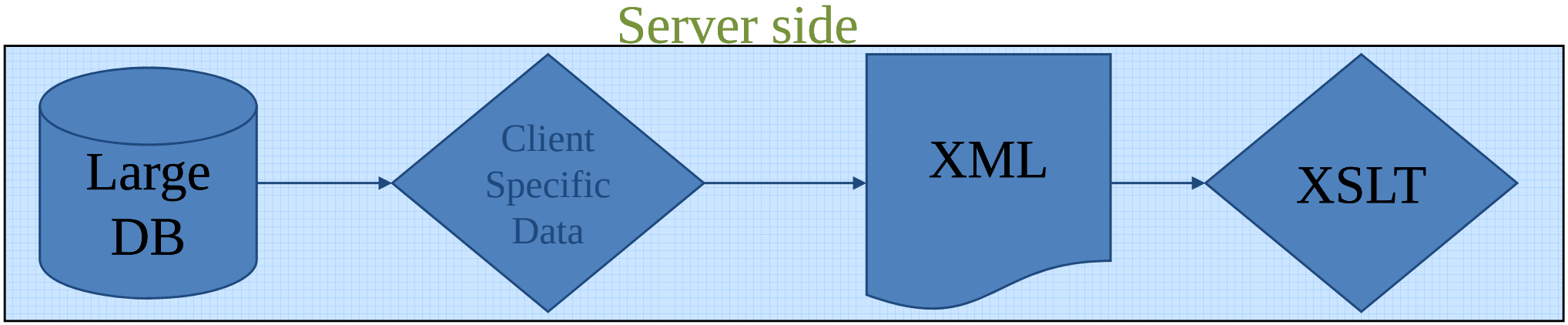
XROSA- B2B XML Concept



Server-side systems and XML

Scenario 1

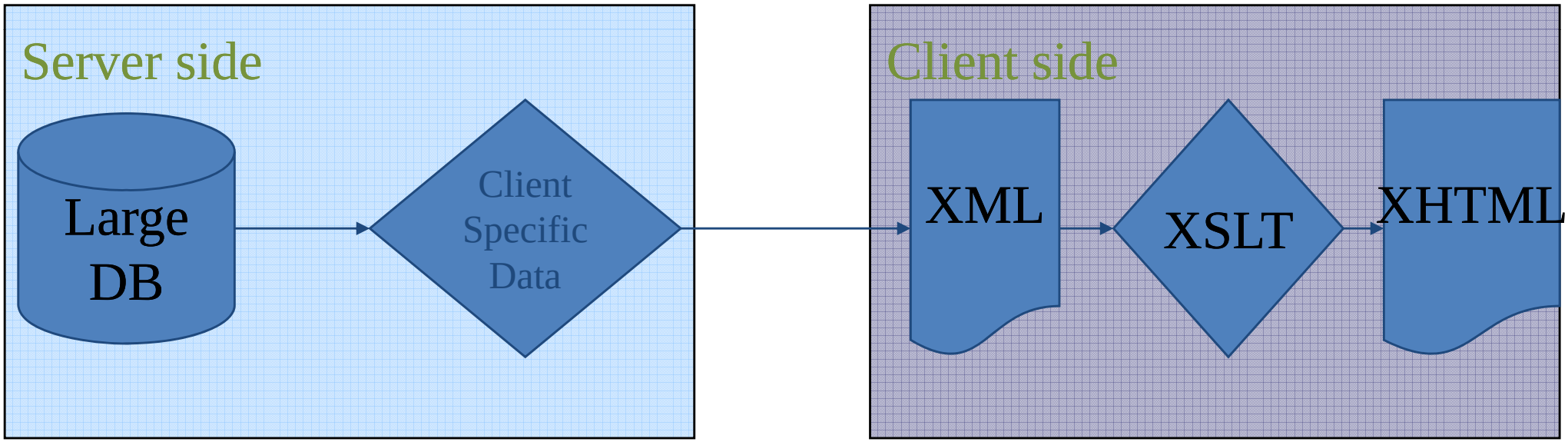
- Large database provides client-specific records
- Data could be exported to XML document on server-side
- Custom XSL and DTD for client's recordset could exist on server



Server-side systems and XML

Scenario 2

- Large database provides client-specific records
- Data could be exported to XML document over the web to client-side
- Custom XSL and DTD for client's recordset could exist on client system (ready to deliver to CD or other client-side technology)



When to Use Server-side and XML technologies

- Good for storing client-specific settings
- Delivering customised data to a client in a useful, easy to manipulate form
- Delivering specific sub-sets of data from much larger data sets (such as delivery country-specific stories from an international news service)
- Essentially, uses and implementation only limited to creativity of developers and support tools (such as XML, XSLT and XSD parsers)
- Server-side scripting gives the management and control functionality, XML gives the delivery and end-user customizability aspects

XML and DBMS interaction

- The DOM provides an interface to the XML document.
  - In Microsoft technology, the XML parser is a COM object and can be instantiated in any tool that uses COM objects.

    Set oXMLDoc = CreateObject (“MSXML4.DOMDocument”)

    oXMLDoc.Load (“c:\mvp.xml”)
  - This describes the possibility of programmatic access to XML data using Visual Basic, C++, Java or Javascript.
  - Thus, XML data can be readily imported to databases as well as exported.
  - The accessibility of XML data is fuelling a revolution known as Web Services.

Remote Communications and Messaging

- Entering the age of Web Services
- Discoverable services
- Self-describing services
- Distributed architectures
- Development moving away from Desktop or Web, rather Objects and Applets

What are Web Services?

- Interoperability, Interoperability, etc.
  - Make use of other people's code
  - Text (XML) based approach
- Scalability
  - Providers: Build and publish new services
  - Clients: Find and use required services
- According to Microsoft (2003) Web Services:
  - Allow applications to share data
  - Are discrete units of code; each handles a limited set of tasks
  - Are based on XML, the universal language of Internet data exchange
  - Can be called across platforms and operating systems, regardless of programming language.

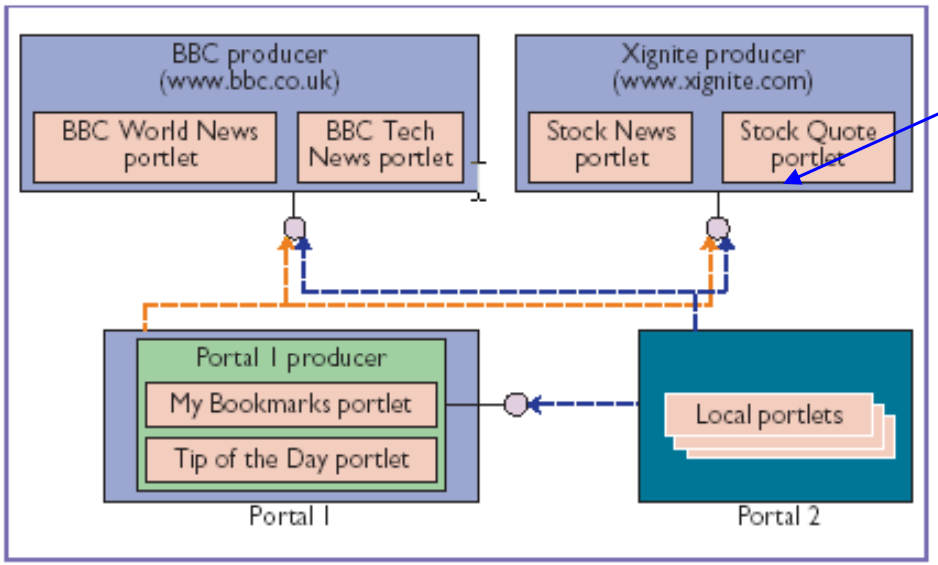
(Wilczynski, David, 2004)

What are Web Services?

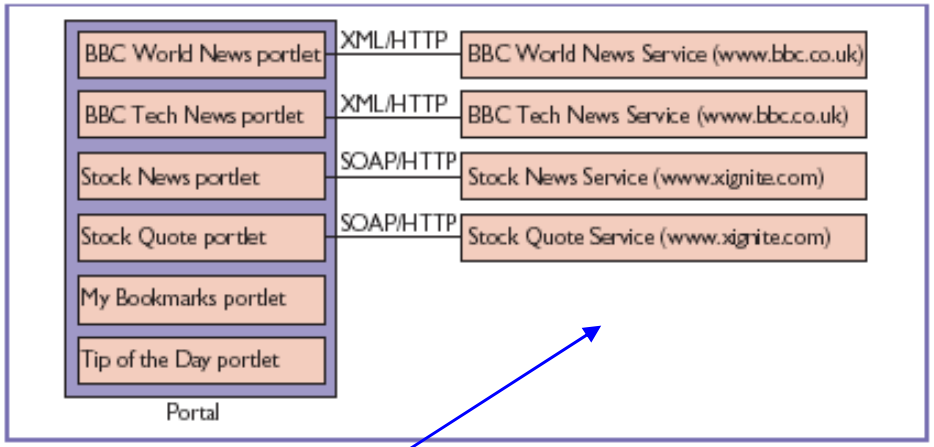
- The benefits of Web Services (ibid) include:
  - Opening the door to new business opportunities by making it easy to connect with partners;
  - Delivering dramatically more personal, integrated experiences to users via the new breed of smart devices including PCs;
  - Saving time and money by cutting development time;
  - Increasing revenue streams by easily making your own Web services available to others.

WSDL & Portlets

- In first generation portal design, a portlet is a software component local to the portal, with either its own content or content provided by an external service.
- In a leading edge second generation approach (Bellas, 2004, p.55) each remote portlet producer implements a standard Web service interface (defined in Web Service Definition Language) through which portals can interact with the remote producer's portlets. Portals can consume a given remote portlet, and portals can export their local portlets to other consumers.

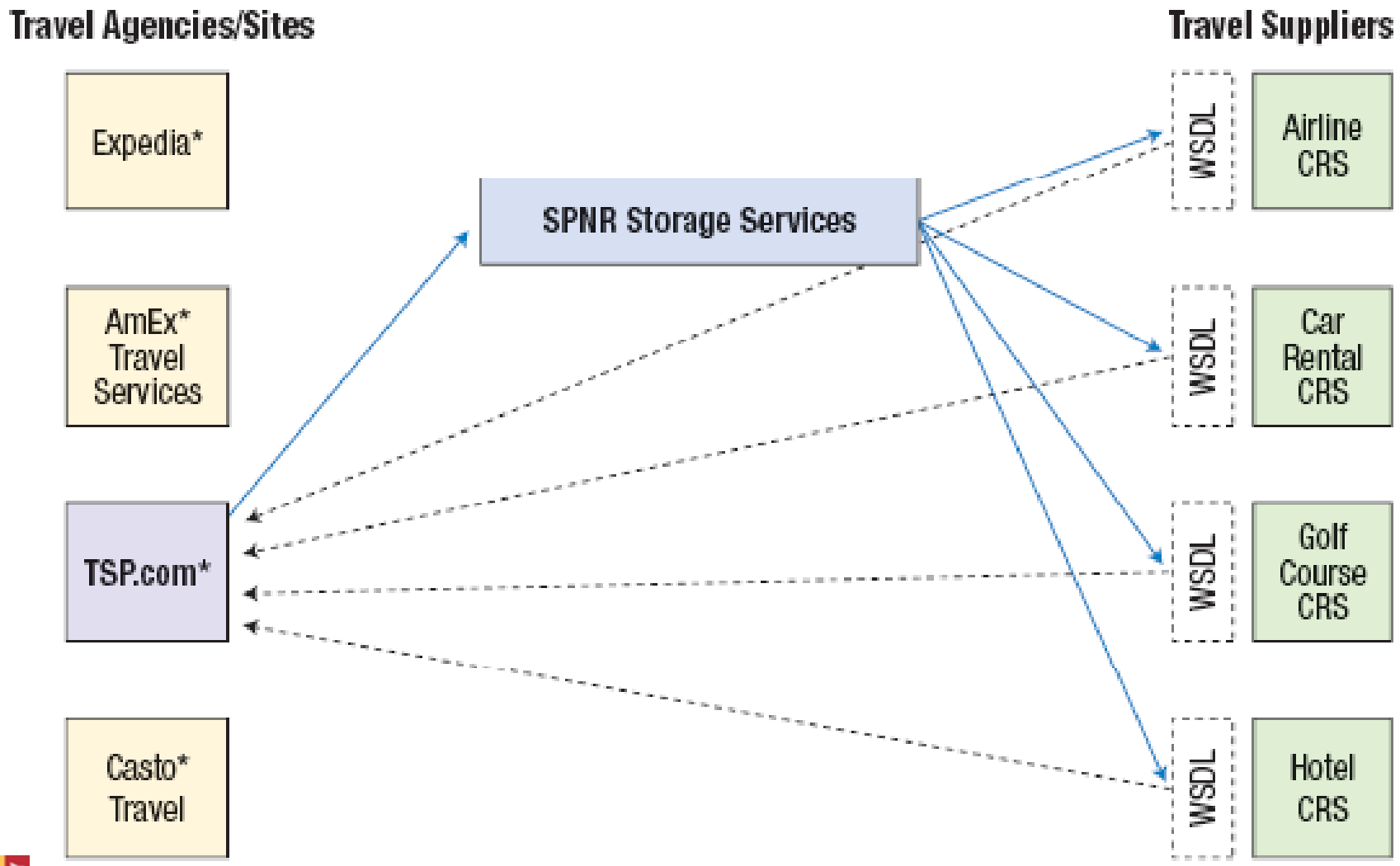


Portlets are exportable and can be remotely consumed



Portlets are local

Web Services in Action



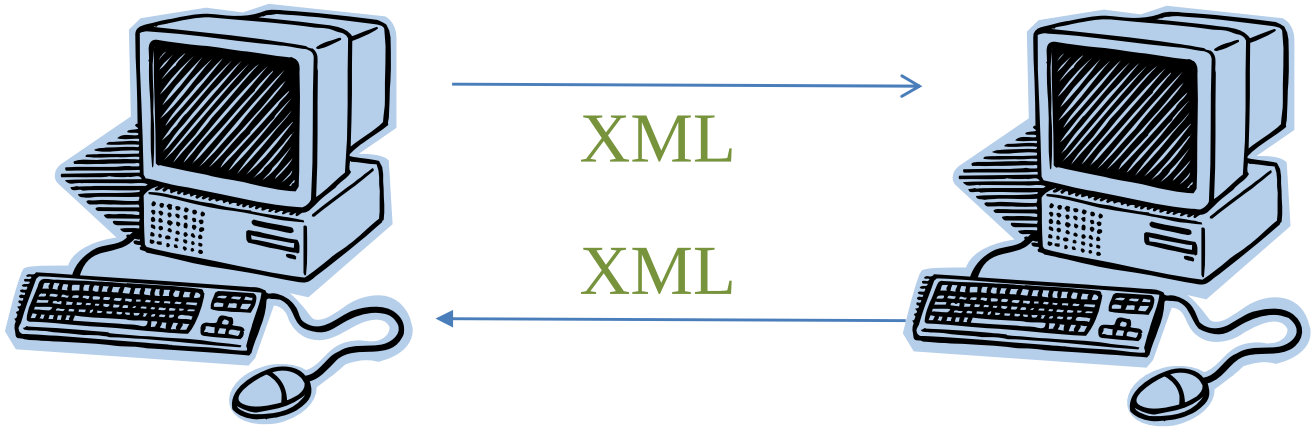
Ref: Data Dimension (2002)

Web Services Overview

- An platform and language independent architecture for program to program communications
- A Web Service is a software component that can be:
  - *Described* using a service description language
  - *Published* to a registry of services
  - *Invoked* through a declared API (usually over the network)
  - *Composed* with other services

(ibm, 2004)

A Basic Web Service



Computer A:
Language: Perl
Operating System:
Windows 2000

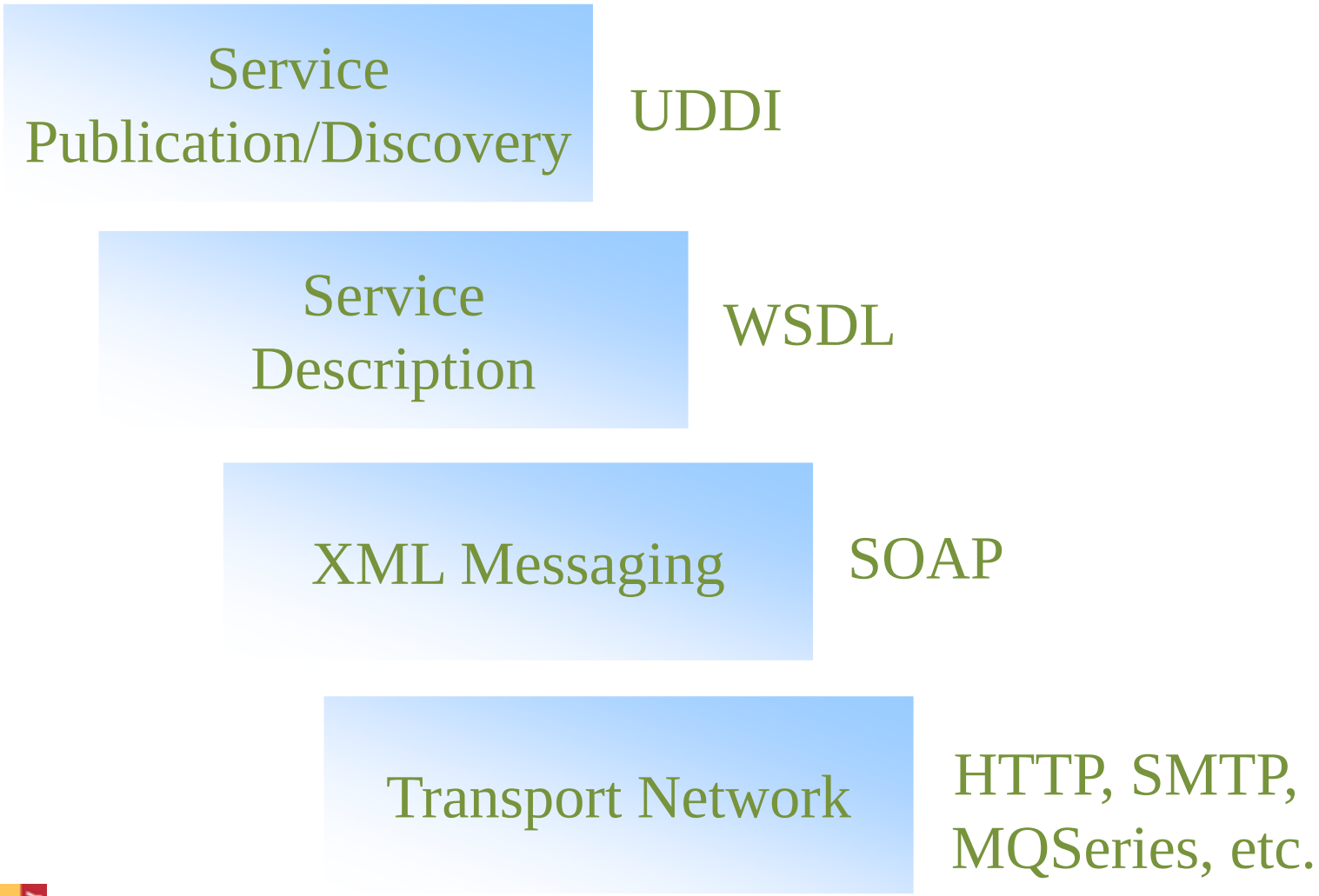
Computer B:
Language: Java
Operating System:
Linux

Requirements of a Service?

- Clients must be able to discover services. (Universal Description Discovery and Integration - UDDI)
- Providers must be able to describe their service (Web Services Description Language – WSDL)
- Providers must be able to describe the interactions (conversations) to use the service (Web Services Conversation Language - WSCL)
- This is like tuning into different stations on a radio – the music and station may be different but the radio still knows how to play it

(Wilczynski, David, 2004)

The Web Services Stack



(ibm, 2004)

WSDL

- WSDL: Web Service Description Language.
- WSDL is an XML grammar for specifying an interface for a web service.
- Specifies
  - location of web service
  - methods that are available by the web service
  - data type information for all XML messages
- WSDL is commonly used to describe SOAP services.

WSDL In a Nutshell

<definitions>: Root WSDL Element

- **<types>: What data types will be transmitted?**
- **<message>: What messages will be transmitted?**
- **<portType>: What operations (functions) will be supported?**
- **<binding>: What SOAP specific details are there?**
- **<service>: Where is the service located?**

WSDL Excerpt: Weather Service

```
<message name="getWeatherRequest">
  <part name="zipcode" type="xsd:string"/>
</message>
<message name="getWeatherResponse">
  <part name="temperature" type="xsd:int"/>
</message>

<portType name="Weather_PortType">
  <operation name="getWeather">
    <input message="tns:getWeatherRequest"/>
    <output message="tns:getWeatherResponse"/>
  </operation>
</portType>
```

WSDL Excerpt: Weather Service

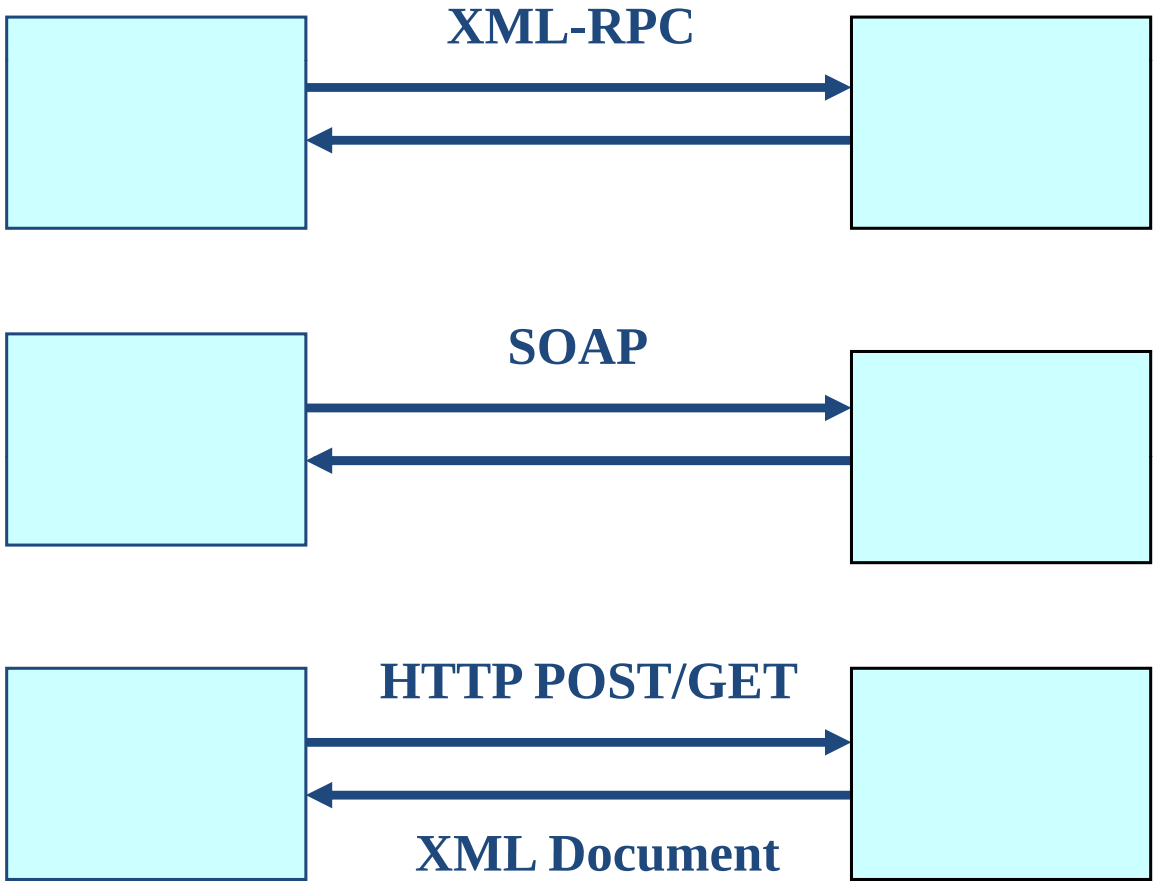
```
…
 <service name="Weather_Service">
     <documentation>WSDL File for
    Weather Service</documentation>
     <port binding="tns:Weather_Binding"
          name="Weather_Port">
          <soap:address
          location="http://ecerami.com/soap/servlet/rpcrouter"/>
     </port>
  </service>
</definitions>
```

So What?

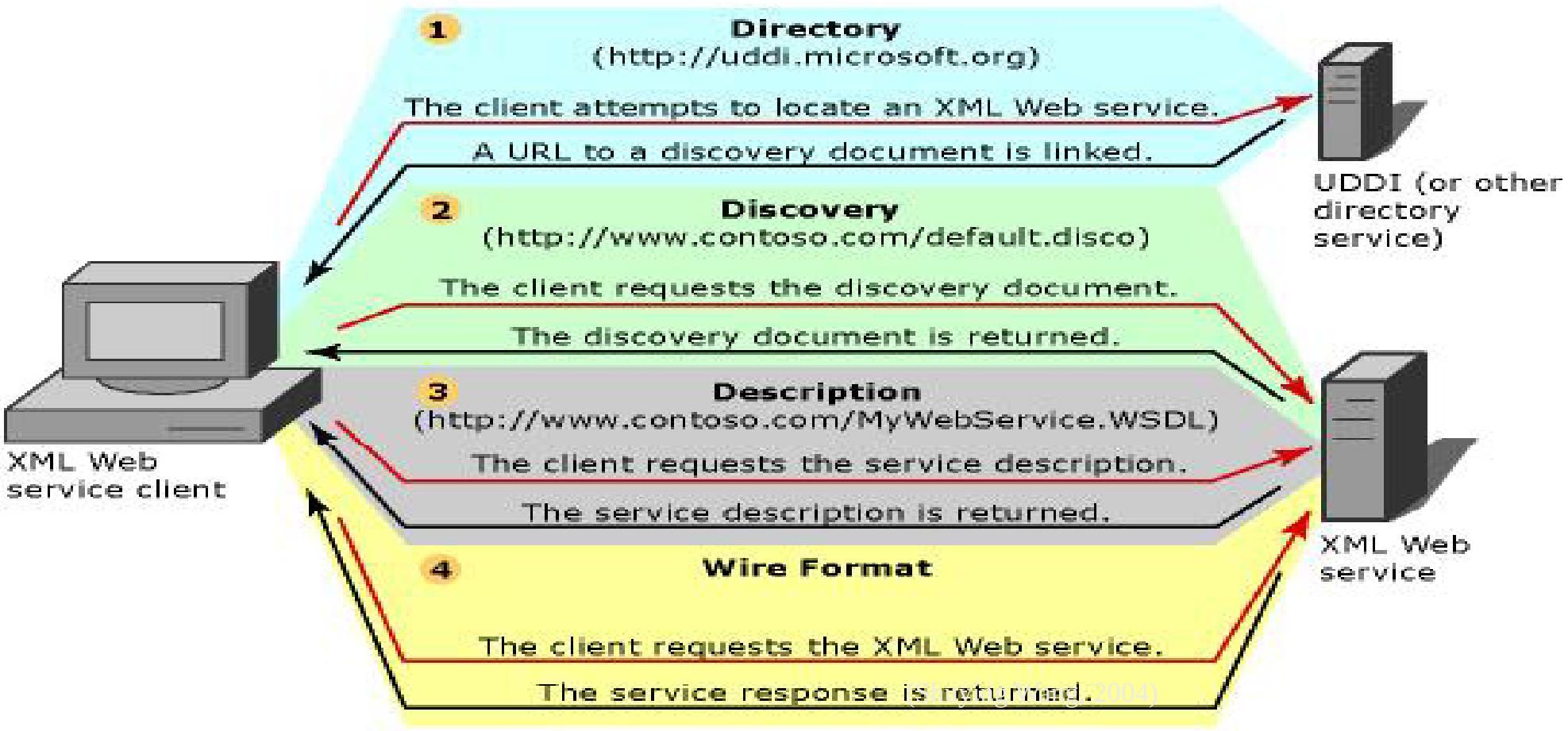
- Given a WSDL file, a developer can immediately figure out how to connect to the web service.
- Eases overall integration process.
- Better yet, with WSDL tools, you can *automate* the integration…

XML Messaging

- There are several alternatives for XML messaging:
  - XML Remote Procedure Calls (XML-RPC)
  - SOAP
  - Regular XML transported over HTTP
- Any of these options are valid.



Architecture of Web Service



Web Services: Self Describing

- If you publish a new web service, you should also publish a public interface to the service.
- At a minimum, you should include human-readable documentation so that others can easily integrate your service.
- If you have created a SOAP service, you should also include a public interface written in a common XML grammar.

Web Services: Discoverable

- If you create a web service, there should be a relatively simple mechanism to publish this fact.
- Likewise, interested parties should be able to easily discover your service.
- The discovery service could be completely decentralized or completely centralized.

Using the Protocols Together

Step 1: **Find Services via UDDI**

Step 2: **Retrieve Service Description File: WSDL or XML-RPC Instructions**

Step 3: **Create XML-RPC or SOAP Client**

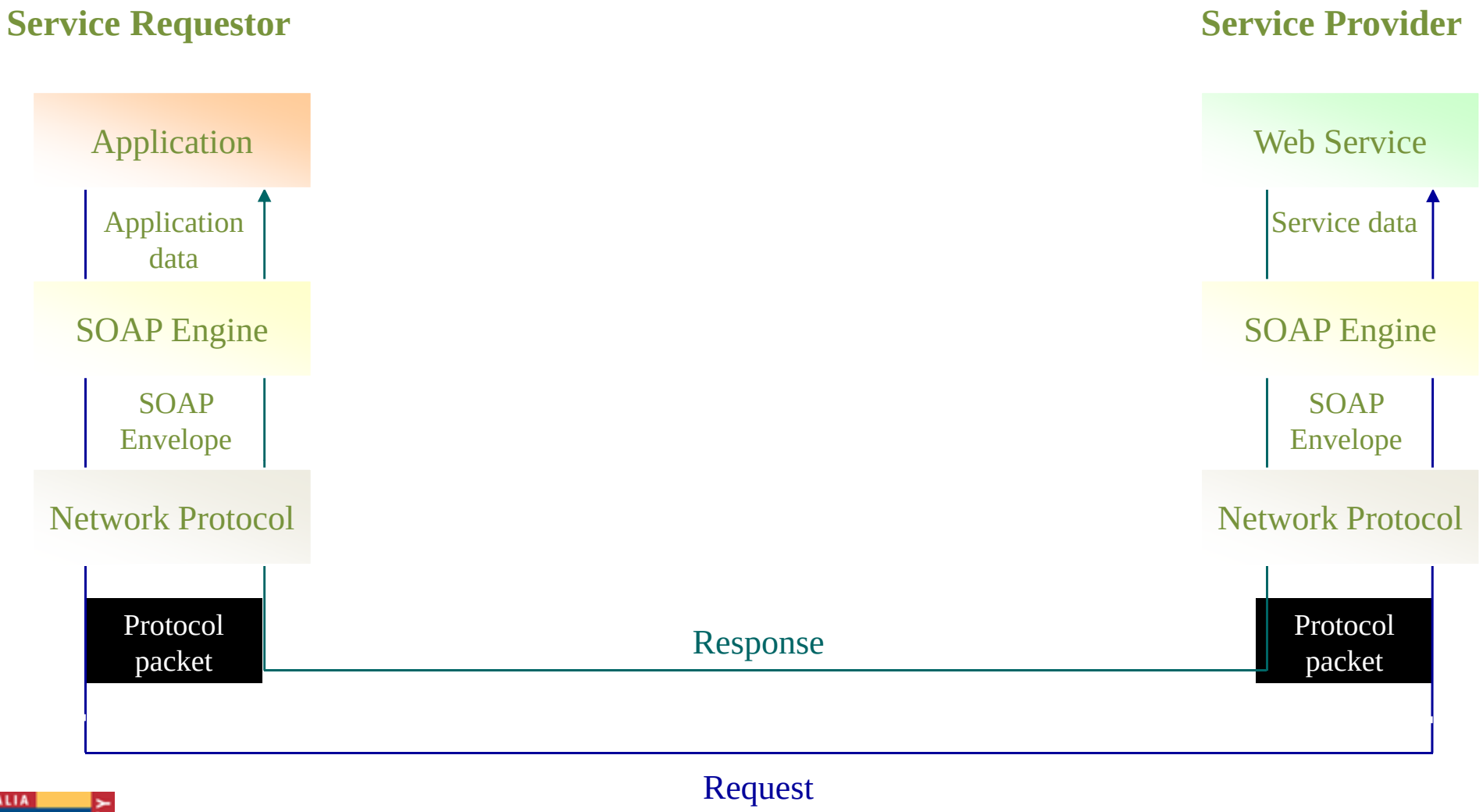
Step 4: **Invoke Remote Service**

What is SOAP?

- Simple Object Access Protocol
- SOAP version 1.2 is a lightweight protocol for exchange of information in a decentralized, distributed environment
- It is an XML based protocol that consists of four parts:
  - an envelope that defines a framework for describing what is in a message and how to process it,
  - a transport binding framework for exchanging messages using an underlying protocol,
  - a set of encoding rules for expressing instances of application-defined data types and a convention for representing
  - remote procedure calls and responses. Part 1 (this document) describes the SOAP envelope and SOAP transport binding framework;

[http://www.w3.org/TR/soap12-part1/ (1.2 Working Draft)]

SOAP: XML Messaging



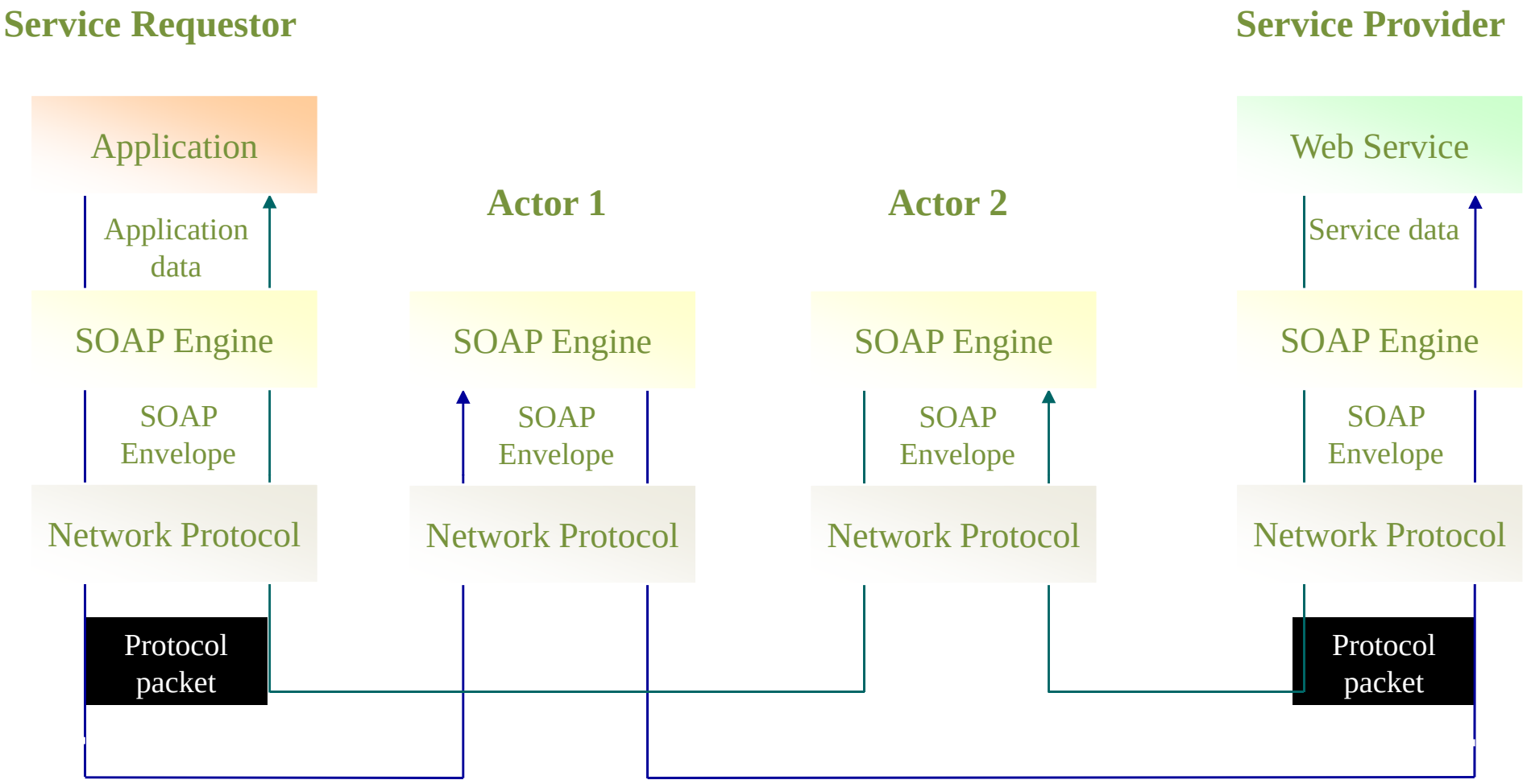
(ibm, 2004)

SOAP Intermediaries

- Vertical extensibility
  - Capability of adding new pieces of information within a SOAP message
- Horizontal extensibility
  - Targeting different parts of a SOAP message to different recipients
  - Achieved through intermediaries that process parts of SOAP messages as they travel from source to destination
    - Security
    - Scalability
    - Tracing
  - SOAP headers can be explicitly targeted at intermediaries
    - SOAP-ENV:actor attribute
    - http://schemas.xmlsoap.org/soap/actor/next

(ibm, 2004)

SOAP Intermediaries



(ibm, 2004)

SOAP in Action

- Generic XML-based RPC protocol
- Initial specification only covers HTTP-binding
- Use of XML Schema
- Multiple protocol bindings available today(SMTP, FTP, JMS, etc.)
- RPC-style and document-style communication

Example SOAP Message Request

- A standard XML-based format to describe a SOAP request for a Web Service
- Provides all the information required by the Web Service provider to process the request
- General format of a SOAP request:

```
HTTP Header
SOAP Action
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
     <!-- Soap Header is optional -->
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
     <!-- Serialized method invocation data -->
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

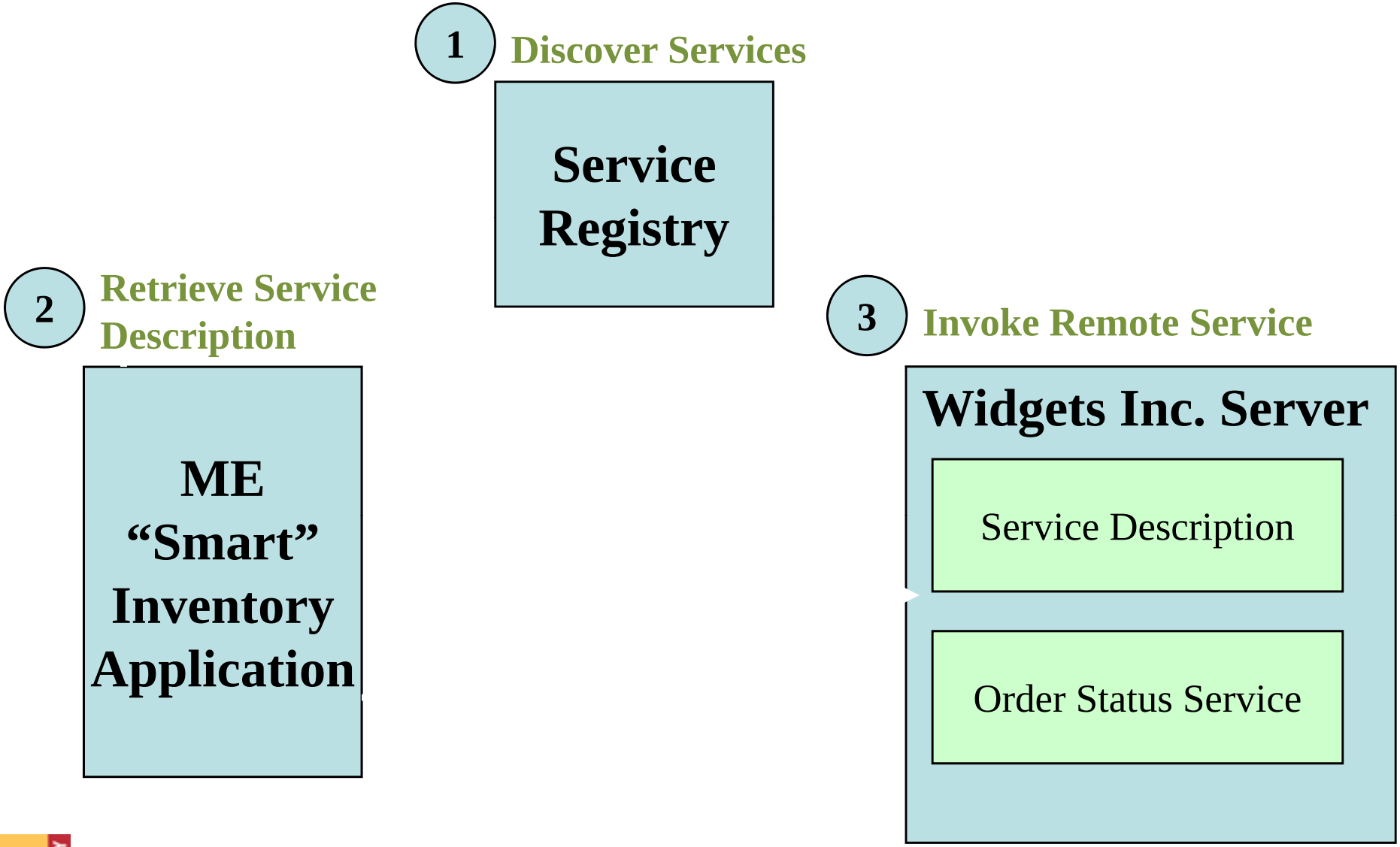
Example SOAP Message Response

- A standard XML-based format to describe the Response generated by a Web Service
- Contains information that is to be passed back to the client
- General format of a SOAP response:

```
HTTP Header

<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
     <!-- Serialized Response Data -->
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Just-In-Time Integration



Hype v. Reality

- How close are we to creating “Just-in-time” integration?
- Currently, only some processes can be automated:
  - automatic registry query
  - automatic invocation of service
- However,
  - no mechanism exists for automating business relationships
  - no mechanism exists for evaluating the quality of services

Industry Landscape

- Many companies are investing heavily in web services
- Currently many competing frameworks for building web services
- Three main contenders:
  - Microsoft .NET
  - IBM Web Services
  - Sun Open Net Environment (ONE), Java
- All frameworks share a commitment to the same web services standards

The Future

- Who knows
- Very hard to predict
- Most likely distributed app based
- Remotely callable components
- Heavy reliance on Web Services
- Distinction between client / server functionality may become less relevant
- Perhaps true platform independence