Emerging XML Standards for e-Business: UDDI, ebXML and the W3C

John Ibbotson

XML Technology and Messaging

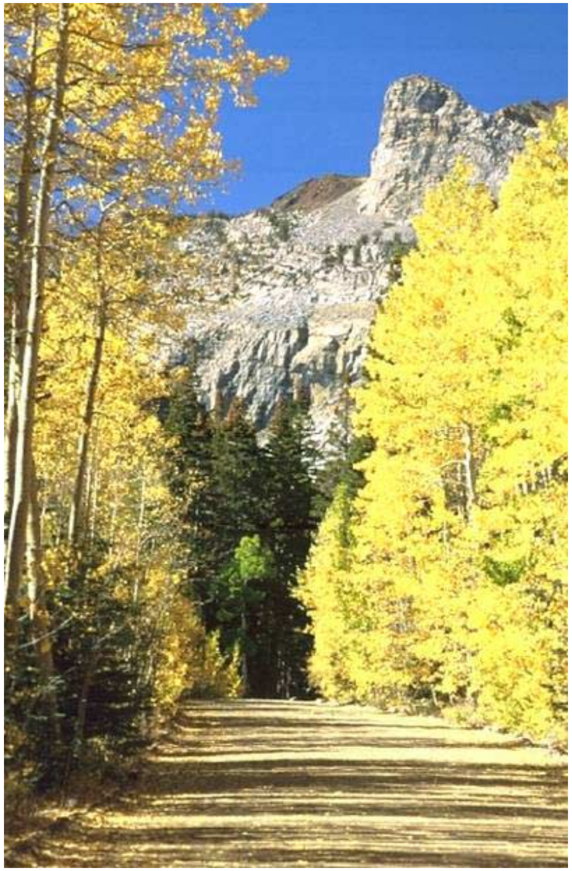
IBM

Agenda

- Who Needs Standards ?
- W3C and the Core Technologies
- ebXML and EDI Convergence
- UDDI and Web Services
- Summary and Questions

The Road to Interoperability

- Vendors developed proprietary solutions.
- Customers needed systems to be connected or realized that no one vendor supplied everything they need.
- Vendors reluctantly decided to work together.
- Integration became a business.
- Vendors decided that standards should be part of the business model.



What does this mean for data ?

- Data has to be able to flow
  - between different applications
  - using different programming environments
  - on different hardware
  - running different operating systems
  - using different communication protocols.
- XML can help you do this.
- A bit of extra work is needed.

Key Web Technologies

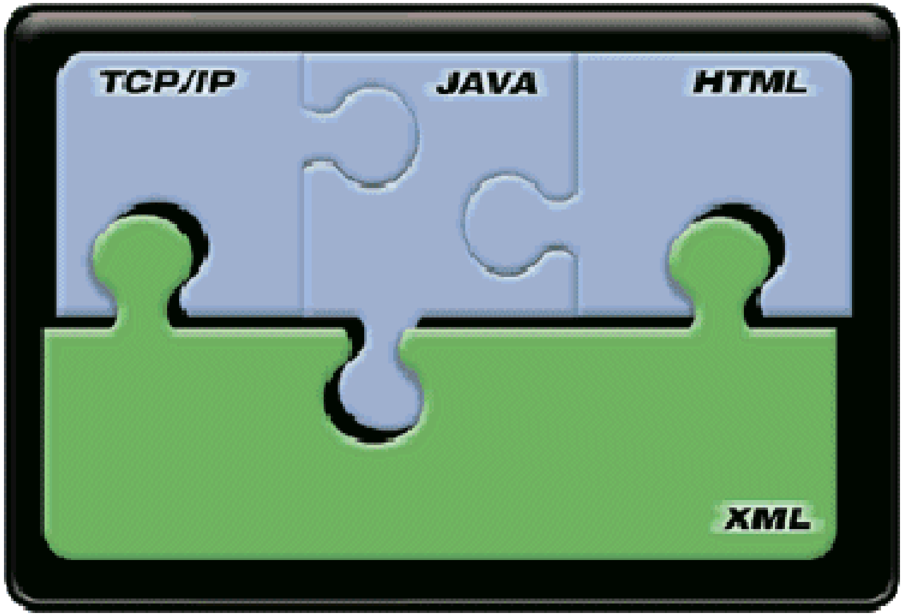
- The web has been built on technologies such as TCP/IP, http, HTML, Java, most of them open standards.

"The worst in our industry - the elements we must leave behind - is the mentality that seeks to own standards, and establish choke holds based on proprietary technology. Let’s remind ourselves that the spark that ignited the Internet revolution was not technology. It was an agreement to industry open standards. We have to make sure the underlying information technologies are free of closed, proprietary standards."

- Lou Gerstner, TeleCom '99

XML and Standards

- We **used to say** that XML brings data to the web, adding the necessary component for 90% of e-business applications.



We need more than Data

- What are we doing with XML for e-business?

vertical industry message content

cross-industry reusable components

message envelopes

trading partner agreements

web service descriptions

security

business process modeling

workflow

Co-opetition: Friend or Foe ?

- Why would companies work together when they are competing for the same customers?

- IBM’s philosophy is that we will cooperate on standards development but we will compete (aggressively) on products.

- Alliances shift from time to time and vary by standardization effort.

What is the Motivation ?

- Get more companies involved in global e-business.
- Decrease procurement costs and become more efficient organizations.
- Manage the growth of B2B early.
- Increase potential trading partners by being able to find them and then using the same commerce protocols.

- Move from expensive older e-commerce technologies to newer ones that take advantage of the Internet.

Is EDI Dying ?

- Large companies don’t believe so.
- Outside the Fortune 1000, only about 5% of the small- and medium-sized enterprises have signed up.
- “EDI” comes in several cross-industry and vertical industry flavors.
- The rigidity of the message sets, expensive transactions, and slow standardization processes make a bad or incomplete fit for the fast-changing Internet business world.

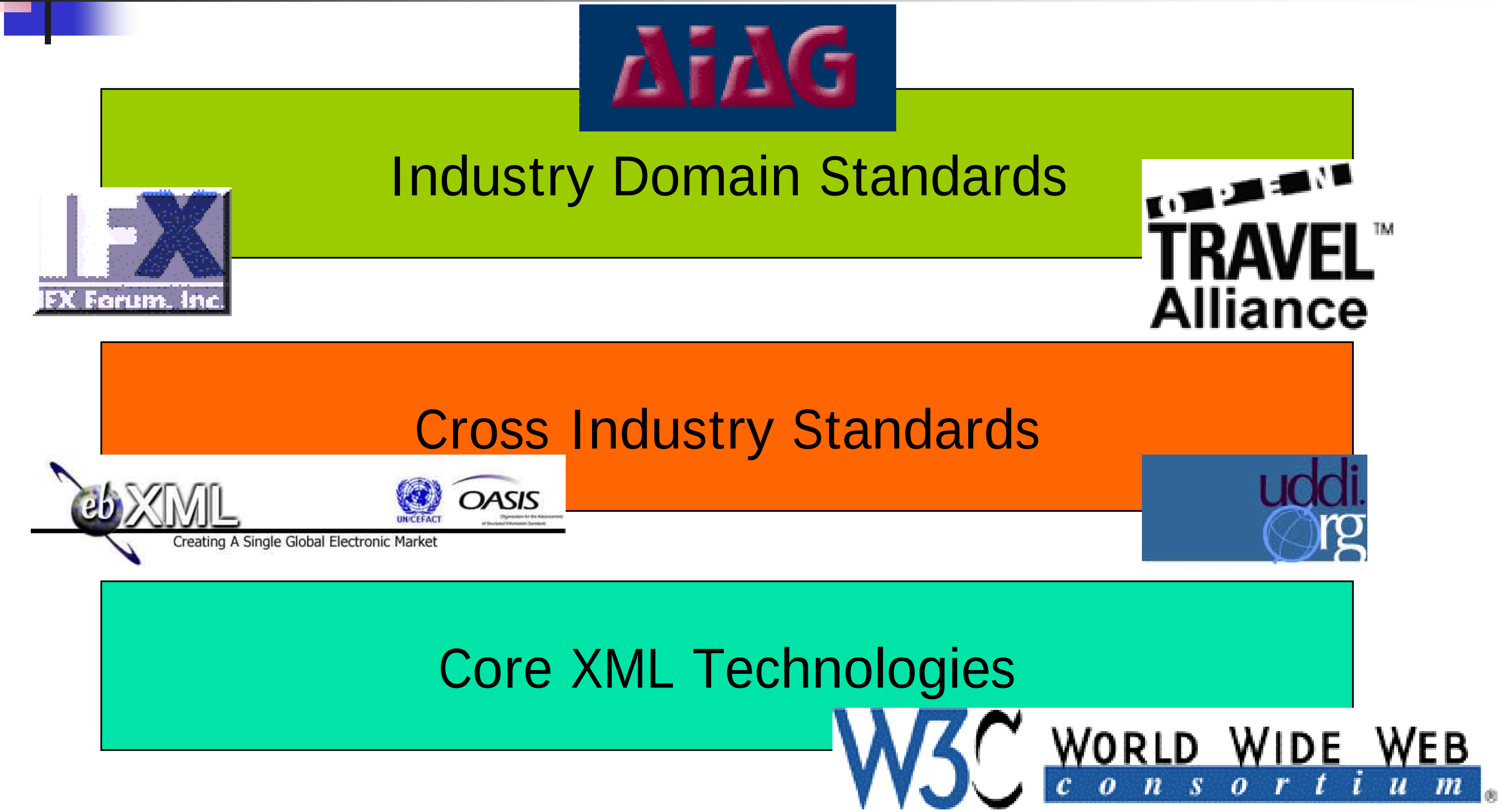
Is XML the Salvation ?

- Companies such as Ariba and Commerce One are using XML to create new e-commerce solutions and marketplaces.
- XML specifications are being developed in many industries.
- Some industries are “re-inventing the wheel” when it comes to B2B XML infrastructure, rather than concentrating on what they understand best.

Many forms of Standardisation

ebXML
Creating A Single Global Electronic Market
UN/CEFACT
OASIS
Organization for the Advancement of Structured Information Standards
IFX
IFX Forum, Inc.
W3C
WORLD WIDE WEB
consortium
®
AIAG
OPEN
TRAVEL
Alliance
TM
uddi.org
Accredited Standards Committee X12
XAML

The Standards Stack



Industry Domain Standards

Cross Industry Standards

Core XML Technologies

World Wide Web Consortium

W3C XML Technologies

- "Recommended" by W3C:
  - XML Specification 1.0: syntax, DTDs
  - DOM Specification 1.0 & 2.0: Document Model
  - XSLT Specification 1.0: transforming XML
  - XPath Specification 1.0: queries, addressing XML docs
  - XHTML Specification 1.0: HTML in XML form
- Works in progress:
  - XML Schema: big improvements over DTDs
  - XSL Formatting Objects
  - DOM 3.0
  - XML Query: a more powerful query mechanism
  - XPointer, XLink
- Other standards:
  - SAX 2.0 (defacto standard, not from W3C)
  - SOAP (proposed standard)

XML 1.0 Specification

- Originally published: February 1998
- In about 35 pages:
  - XML syntax details
  - Document Type Definition (DTD)
- XML 1.0 Specification, Second Edition
  - errata applied to original spec, not a new version
  - now a "recommendation" (replaces Feb 1998 edition)
  - http://www.w3.org/TR/2000/REC-xml-20001006
- Supplementary specs:
  - Namespaces in XML (January, 1999)
  - Stylesheet linking (June, 1999)
  - others in progress (XBase, XInclude, Canonical, ...):
  - see http://www.w3.org/XML/Activity.html#future

XML Schema Specifications

- A greatly improved vocabulary definition language
  - Replaces DTDs (superset of DTDs)
  - XML syntax
  - Rich type support
- W3C Working Drafts: http://www.w3.org/XML/Schema
  - XML Schema Part 0: Primer
  - XML Schema Part 1: Structures
  - XML Schema Part 2: Datatypes
- Almost W3C Candidate Recommendation
- 90-95% implemented in Xerces-Java
  - xml.apache.com, XML4J on www.alphaworks.ibm.com

DOM 1.0 Specification

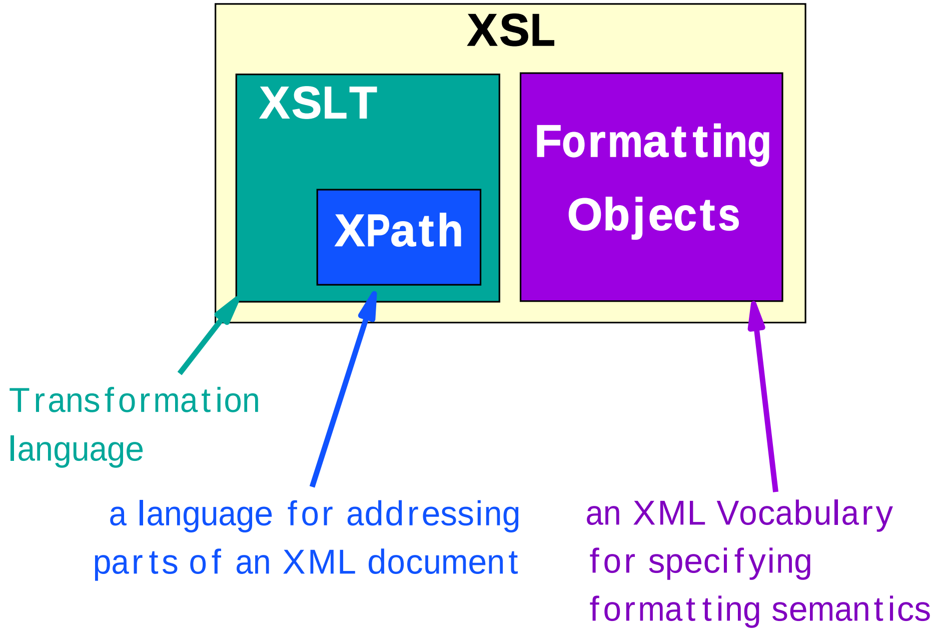
- Models a tree representation of an XML document
  - Tree is created as a result of parsing a document
  - Supports both XML and HTML
- A language-independent object definition and API
  - Bindings for Java in Appendix
- DOM 1.0 W3C Recommendation: October, 1998
  - spec: http://www.w3.org/TR/REC-DOM-Level-1/
- DOM 2.0 recently became a W3C Recommendation
  - New methods, types, interfaces
  - Traversals, namespaces, event model, stylesheets
- DOM 3.0 is currently a Working Draft

SAX 2.0 Specification

- A de-facto "standard" by Dave Megginson
  - Not from W3C
- A free API for event-based XML parsing
  - Instead of getting a complete DOM tree, you get notifications of the arrival of each piece
  - Essential when parsing very large documents
- Available for Java, C++, COM, Perl, Python
- Version 1.0 published May, 1998
- Version 2.0 published May, 2000
- SAX 2.0 support is available in Xerces parser
- See http://www.megginson.com/SAX/index.html

XSL: Extensible Stylesheet Language

- See http://www.w3.org/Style/XSL/
- Three parts:



Transformation language

a language for addressing parts of an XML document

an XML Vocabulary for specifying formatting semantics

XSLT 1.0 Specification

- A transformation language for XML documents
  - Styling (rendering to visual form, like HTML)
  - Transformation (vocabulary translation)
  - Can emit XML, HTML, even non-XML formats
- XSLT documents are well-formed XML
- W3C Recommendation: November, 1999
  - Spec: http://www.w3.org/TR/xslt
- XSLT 1.0 implementations
  - Apache Xalan xml.apache.org
  - LotusXSL www.ibm.alphaworks.com
- XSLT 1.1 and 2.0 are planned

XSL Formatting Objects

- Layout-oriented XML vocabulary
  - Rich representation of documents for printing, various device screens, etc
  - Usually created as output of XSLT
  - Using an appropriate stylesheet
- XSL Specification defines FO's, refers to XSLT
  - Currently W3C Working Draft, last call
  - Coming soon: W3C Candidate Recommendation
  - See http://www.w3.org/TR/xsl/
- FOP open source FO processor implementation (creates PDF) available at xml.apache.org

XPATH 1.0 Specification

- Language for addressing parts of an XML document
  - Used by XSLT and XPointer
  - Basic facilities for manipulation of strings, numbers and booleans
  - Can be used as simple query language
  - Compact, non-XML syntax for use in URIs
- W3C Recommendation: November, 1999
  - See http://www.w3.org/TR/xpath
- XPath 2.0 is planned
- XPath implementation: part of Xalan / LotusXSL, xml.apache.org / www.alphaworks.ibm.com

XML Query Specification

- Query facilities to extract data from real and virtual XML documents
- Relatively new, work in progress:
- Requirements: http://www.w3.org/TR/xmlquery-req
  - Data model: W3C Working Draft: May, 2000
    - Spec: http://www.w3.org/TR/query-datamodel/
  - Query operators: not yet available
  - Query language(s): not yet available
    - May be two: one for human use, another in XML syntax

XLink Specification

- XML elements for links between documents
  - Simple links similar to HTML hypertext links
  - Supports more sophisticated links
- W3C Candidate Recommendation: July, 2000
  - See http://www.w3.org/TR/2000/CR-xlink-20000703/

XPointer Specification

- Fragment indentifier for URI references that locates XML resources
  - Based on XPath
  - Allows for examination of internal document structure and choice based on content
  - Address points and ranges as well as whole nodes
  - Locate information by string matching
- W3C Candidate Recommendation: June, 2000
  - See http://www.w3.org/TR/2000/CR-xptr-20000607.html

XHTML 1.0 Specification

- Reformulation of HTML 4.01 as XML
  - Documents must be "well-formed" XML
  - Elements and attributes are lower-case only
  - For non-empty elements, end tags are required
  - Empty elements (<br />) allowed
  - Attribute values must always be quoted
  - No attribute "minimization"
- W3C Recommendation: January 2000
  - Spec: http://www.w3.org/TR/xhtml1/

VoiceXML

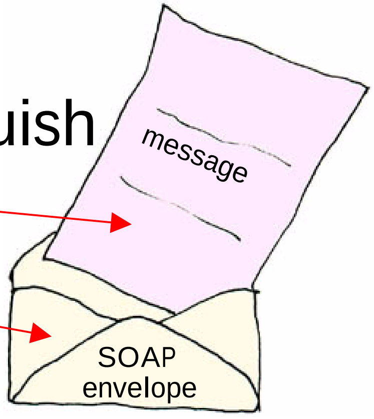
- Designed for creating audio dialogs that feature
  - Synthesized speech, digitized audio
  - Recognition of spoken and DTMF key input
  - Recording of spoken input
  - Telephony
  - Mixed-initiative conversations
  - ...to make Internet content and information accessible via voice and phone
- VoiceXML Forum is an industry organization founded by AT&T, IBM, Lucent and Motorola
- Submitted for consideration by W3C as standard
  - See http://www.w3.org/TR/voicexml/
- Some tools available on www.alphaworks.ibm.com

Simple Object Access Protocol

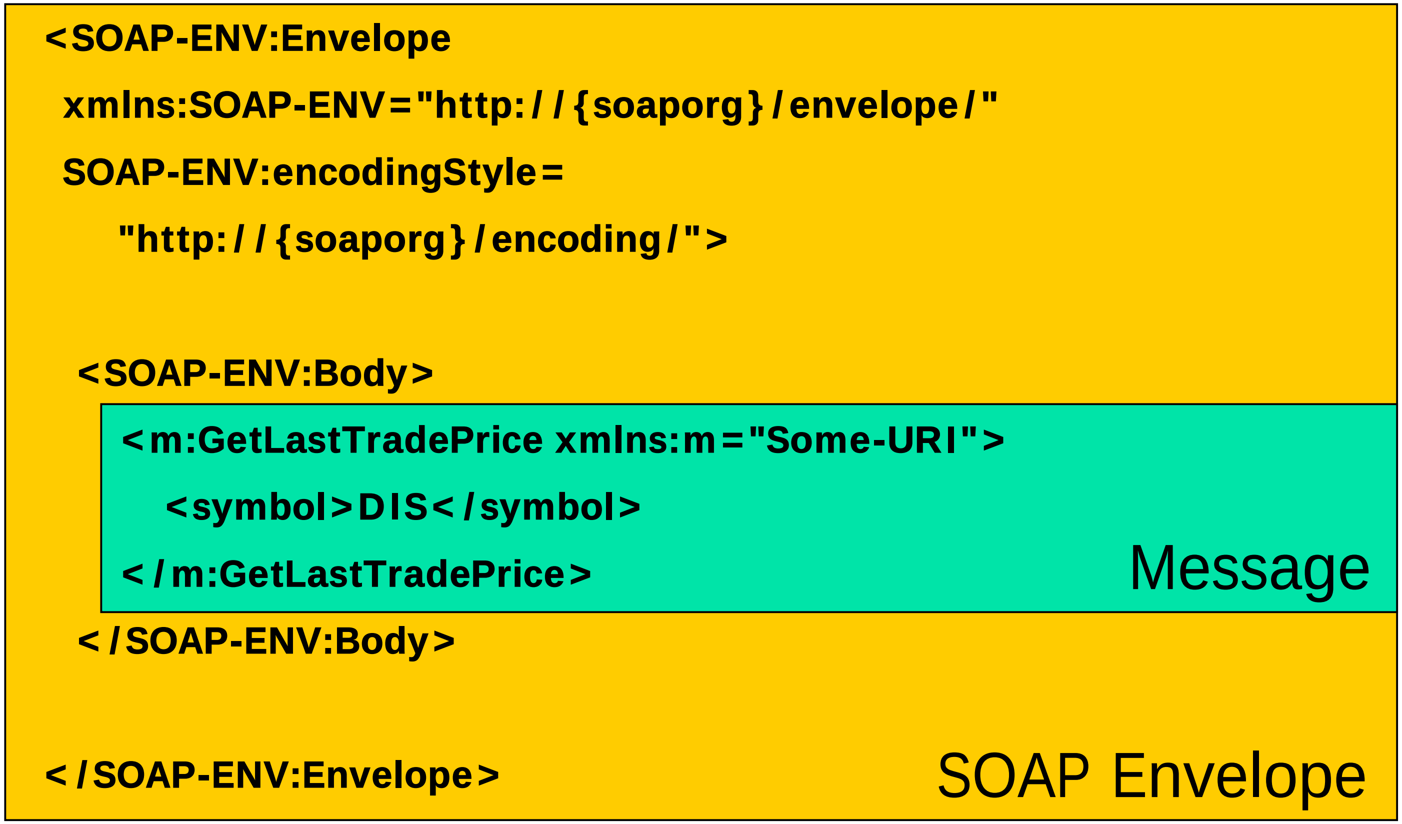
- SOAP 1.0: Userland, Microsoft, DevelopMentor
  - SOAP 1.0 was specific to COM and HTTP
- SOAP 1.1 (April 26, 2000) - includes contributions from IBM and Lotus
  - substitutable Transport bindings (not just HTTP)
  - substitutable Language bindings (e.g. Java)
  - substitutable Data encodings (pluggable)
  - completely vendor-neutral
  - independent of: programming language, object model, operating system, or platform

SOAP Message Structure

- One way message
- Pattern for request/response
  - Invoke a method on a remote object or service
  - Return the result of running the method
- SOAP defines an “envelope”
  - "envelope" wraps the message itself
  - message is a different vocabulary
  - namespace prefix is used to distinguish
    - Application specific vocabulary
    - SOAP Envelope vocabulary



SOAP Request Message



```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://{soaporg}/envelope/"
 SOAP-ENV:encodingStyle=
     "http://{soaporg}/encoding/">

   <SOAP-ENV:Body>
     <m:GetLastTradePrice xmlns:m="Some-URI">
         <symbol>DIS</symbol>
     </m:GetLastTradePrice>
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

SOAP Request Message

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://{soaporg}/envelope/"
 SOAP-ENV:encodingStyle=
     "http://{soaporg}/encoding/">

  <SOAP-ENV:Body>
     <m:GetLastTradePrice xmlns:m="Some-URI“ >
        <symbol>DIS</symbol>
     </m:GetLastTradePrice>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

SOAP Envelope Namespace

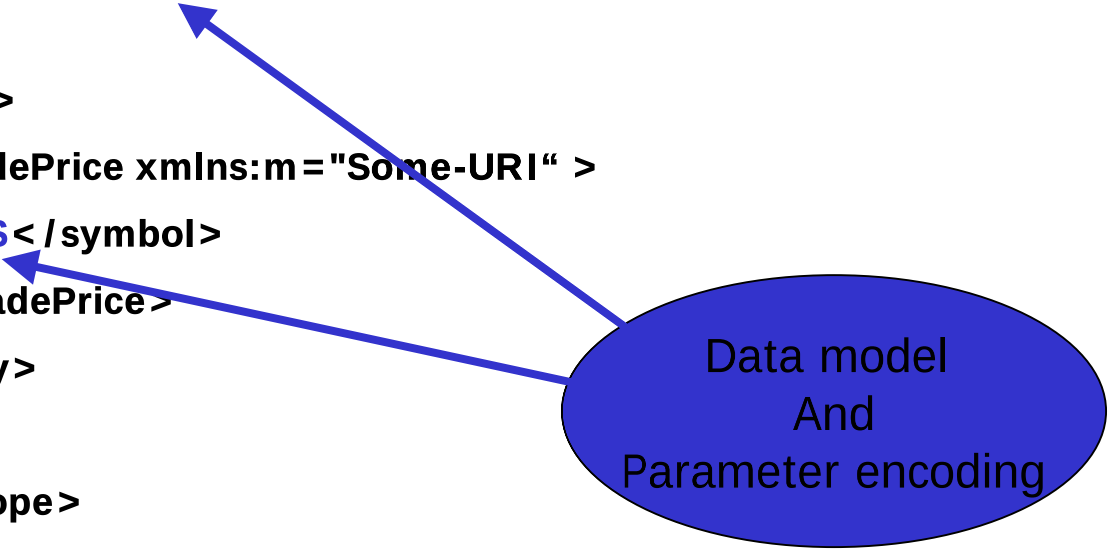
Message Namespace

SOAP Request Message

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://{soaporg}/envelope/"
 SOAP-ENV:encodingStyle=
    "http://{soaporg}/encoding/">

  <SOAP-ENV:Body>
     <m:GetLastTradePrice xmlns:m="Some-URI“ >
        <symbol>DIS</symbol>
     </m:GetLastTradePrice>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```



Data model
And
Parameter encoding

SOAP Response Message

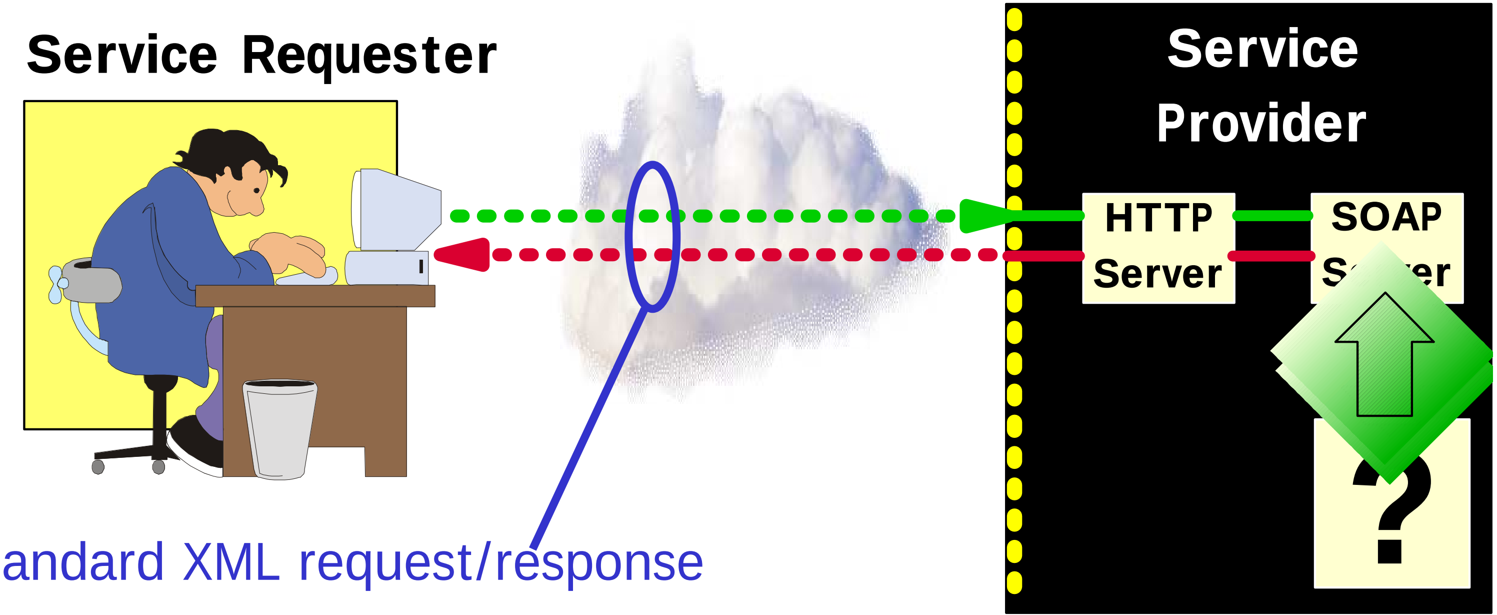
```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://{soaporg}/envelope/"
 SOAP-ENV:encodingStyle=
     "http://{soaporg}/encoding/">

  <SOAP-ENV:Body>
      <m:GetLastTradePriceResponse xmlns:m="Some-URI">
          <Price>34.5</Price>
      </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Result returned
In Body

SOAP hides the Service implementation from the Requestor



standard XML request/response messages

EJB ? CORBA ? COBOL ?

- SOAP envelope
- industry-specific
- message content

Apache SOAP 2.0

- SOAP4J posted to IBM alphaWorks, April 2000
- Contributed by IBM to the Apache Software Foundation, June 2000
- SOAP4J from Apache:
  - Solid SOAP v1.1 implementation, supporting HTTP and SMTP protocols
  - Platform-independent Java
  - Free download from xml.apache.org with source
  - Under development by IBM and others
- SOAP distribution includes:
  - User's Guide
  - API documentation
  - Debugging tool
  - Samples

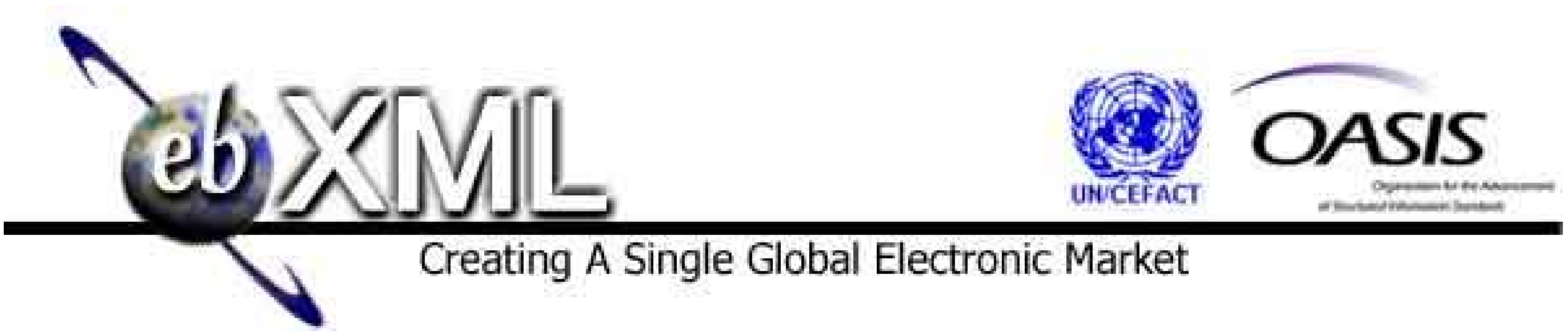
SOAP and Standardisation

- SOAP was submitted to W3C for consideration as a standard (summer 2000)
  - W3C "XML Protocol" activity: Sept 13, 2000
    - work is done in public
    - http://www.w3.org/2000/xp/Group/
    - drafts available: requirements, definitions
    - chair: David Fallside, IBM
    - Web Services workshop scheduled for April 2001
  - Considering SOAP as input to the process, not simply a rubberstamp of SOAP spec
- Industry-specific XMLP messages will start a new round of vocabulary standards work

W3C Summary

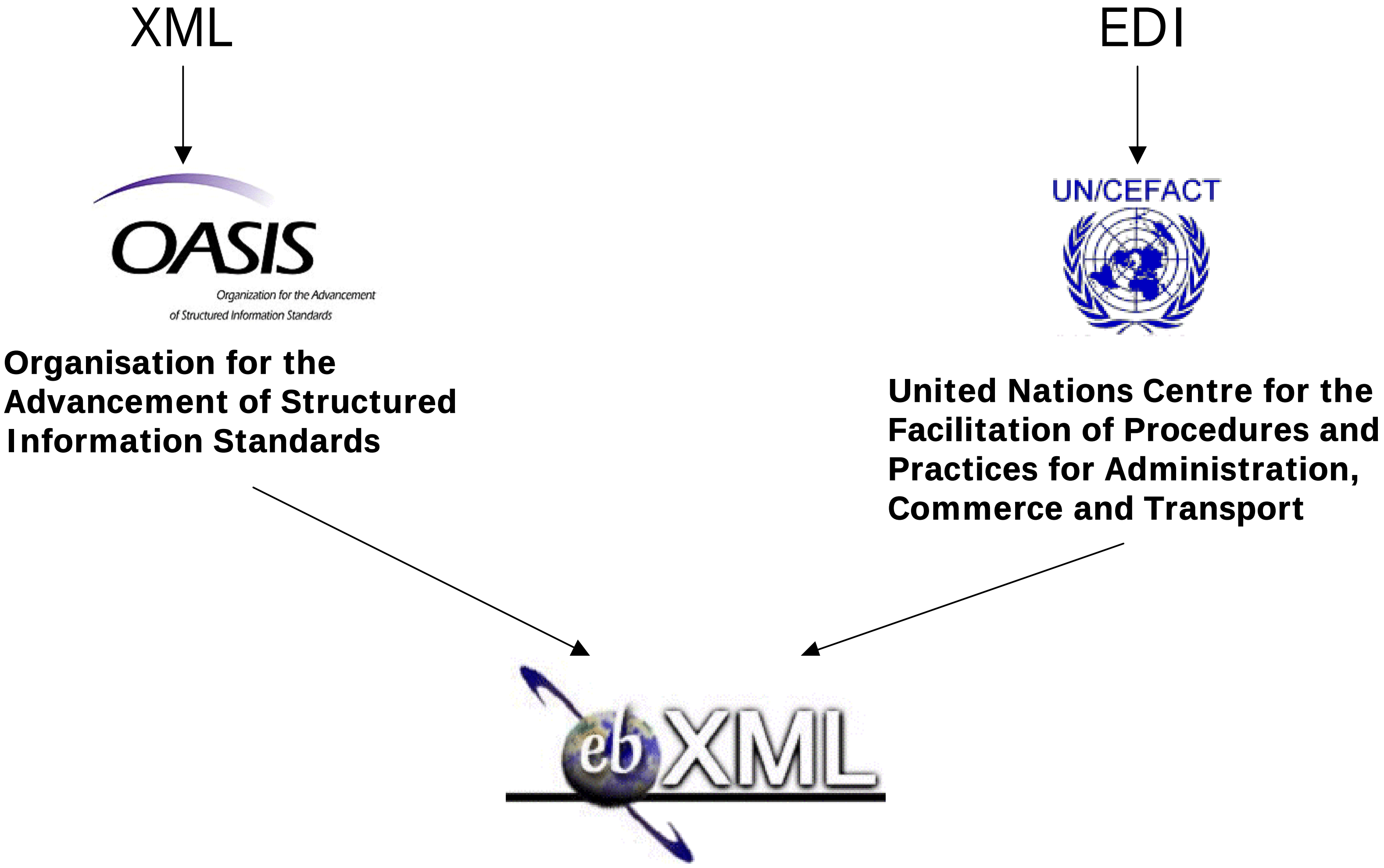
- W3C is the home for XML technology standards
- XML Schema an important standard
  - Still undergoing standards process
  - Some concern over its complexity
- XML Protocol is WG for SOAP
  - Expect other XMLP associated standards

ebXML


ebXML
UN/CEFACT
OASIS
Organization for the Advancement
of Structured Information Standards

Creating A Single Global Electronic Market

Origins of ebXML



XML

EDI

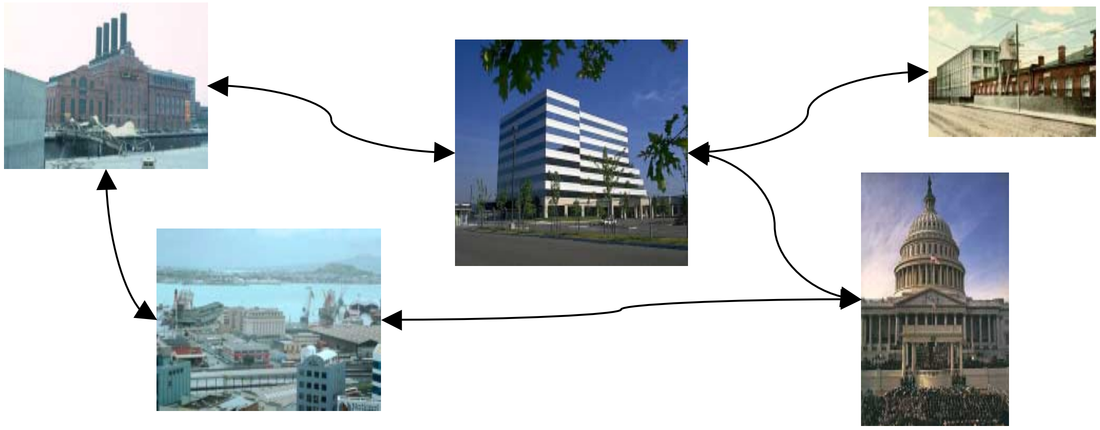
Organisation for the Advancement of Structured Information Standards

United Nations Centre for the Facilitation of Procedures and Practices for Administration, Commerce and Transport

OASIS
Organization for the Advancement of Structured Information Standards
UN/CEFACT
ebXML

The need for ebXML

- ebXML is the joint OASIS - United Nations/ CEFACT “Electronic Business XML initiative.”
- A primary objective of ebXML is to lower the barrier of entry to electronic business in order to facilitate trade, particularly with respect to small- and medium-sized enterprises (SMEs) and developing nations.
- Many vertical efforts are re-inventing infrastructure and not focused completely on the industry vocabulary



ebXML Vision

- *"A single set of internationally agreed upon technical specifications that consist of common XML semantics and related document structures to facilitate global trade."*

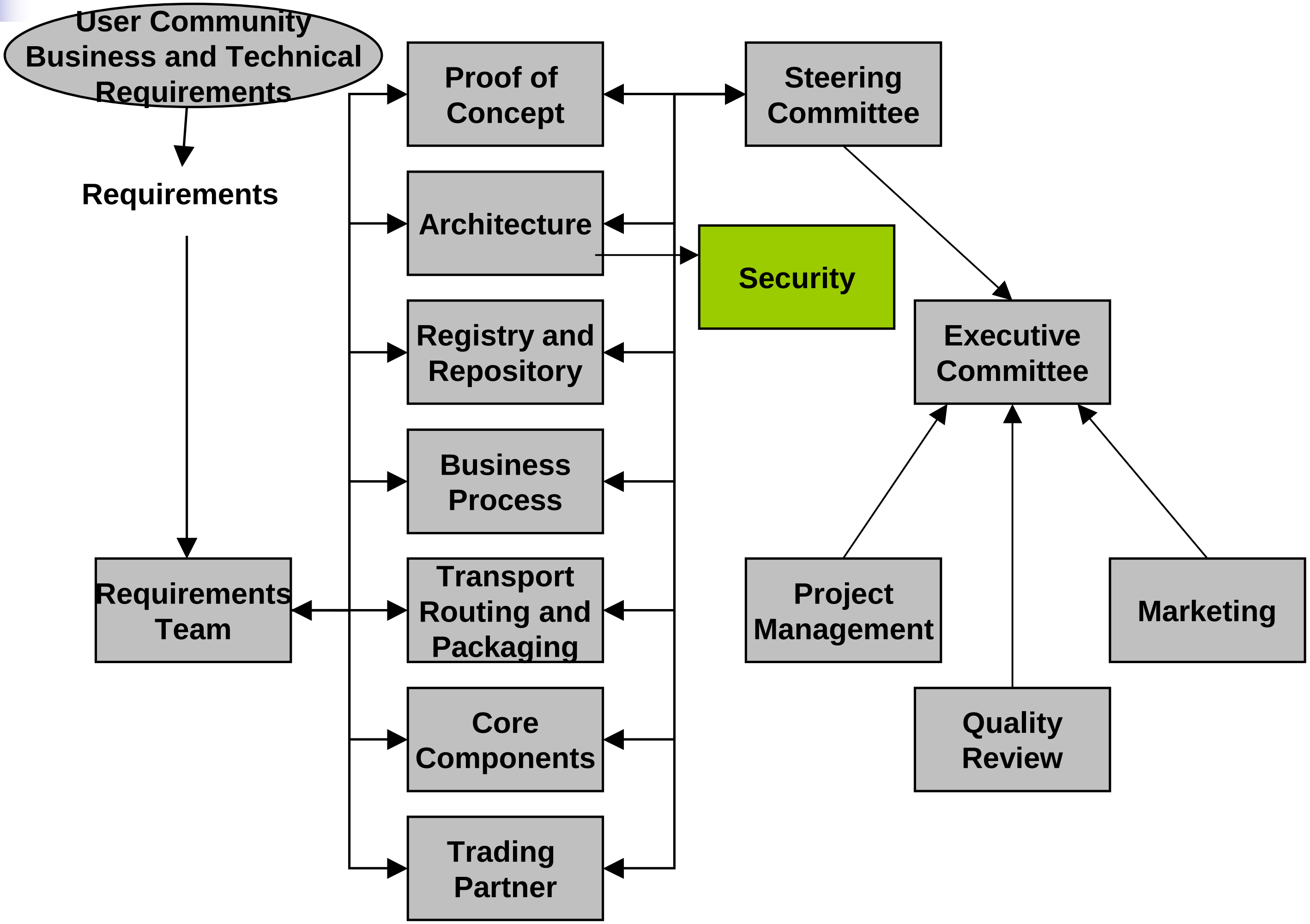
ebXML Operation

- The project lasts 18 months, starting in November, 1999. Ending May 2001
- A key aspect for the success of the ebXML initiative is adherence to the use of the W3C suite of XML and related Web technical specifications to the maximum extent practical.
- Technologies will also be borrowed from other consortiums, as appropriate.

ebXML Objectives

- To maximize interoperability and efficiency while providing a transition path from accredited electronic data interchange (EDI) standards and developing XML business standards, and
- To be submitted to an appropriate internationally recognized standards body for accreditation as an international standard.

ebXML Organization



User Community Business and Technical Requirements

Requirements

Requirements Team

Proof of Concept

Architecture

Registry and Repository

Business Process

Transport Routing and Packaging

Core Components

Trading Partner

Steering Committee

Security

Executive Committee

Project Management

Quality Review

Marketing

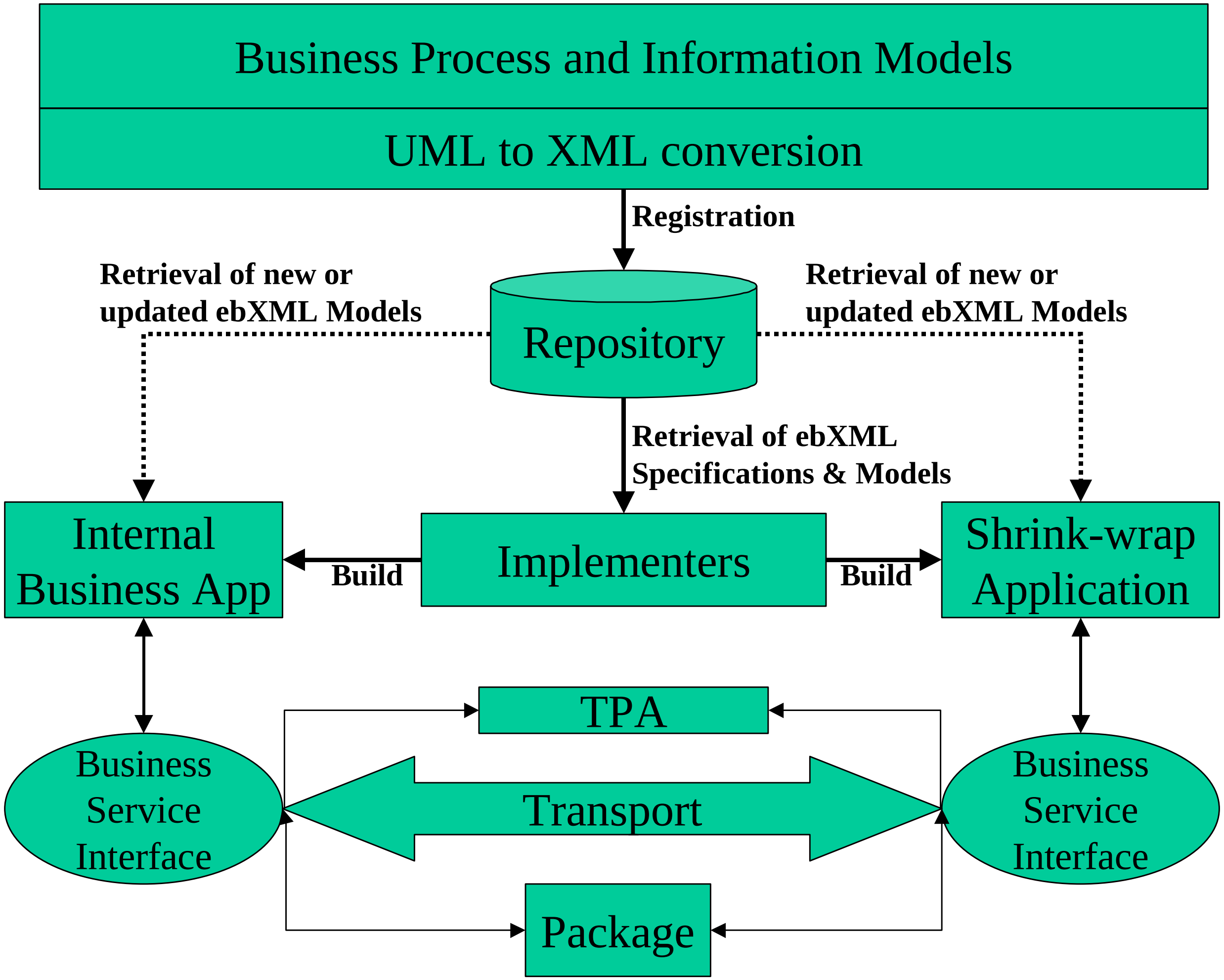
Requirements WG - 1

- Purpose is
  - To provide clearly articulated requirements from representatives of international business and accredited standards organizations to assist the ebXML project team members in developing their deliverables in a consistent manner, and
  - To convey to interested parties the purpose, scope, and vision of ebXML.

Requirements WG - 2

- The business requirements to be addressed by the ebXML initiative are divided into nine core areas
  - General Business,
  - Electronic Business,
  - Globalization,
  - Openness,
  - Usability/Interoperability,
  - Security,
  - Legal,
  - Digital Signature, and
  - Organizational.

ebXML Architecture



Business Process and Information Models

UML to XML conversion

Registration

Retrieval of new or updated ebXML Models

Repository

Retrieval of new or updated ebXML Models

Retrieval of ebXML Specifications & Models

Internal Business App

Build

Implementers

Build

Shrink-wrap Application

TPA

Business Service Interface

Transport

Business Service Interface

Package

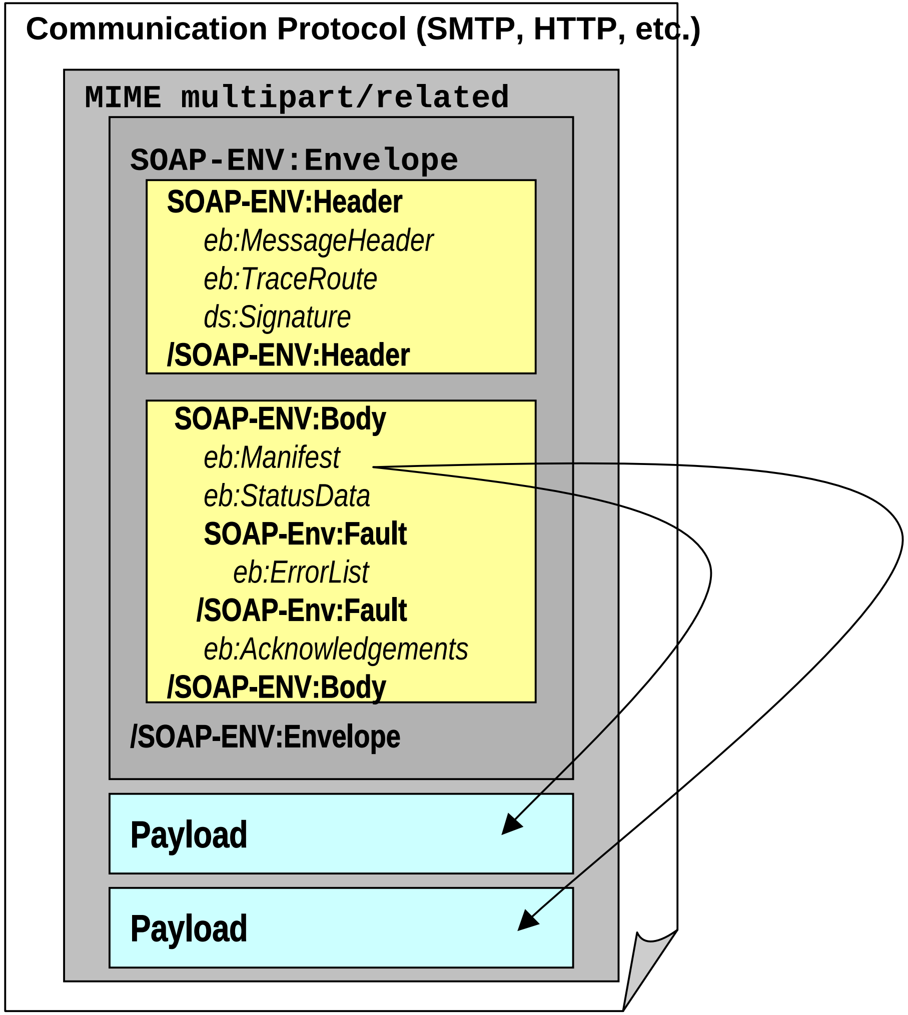
Transport, Routing and Packaging - 1

- Specify how to envelope business documents in regard to
  - related messages in a collection
  - physical and/or logical addressing of destination for messages
- Specify exchange at the application level
- Provide for flexible transaction boundaries
- Provide for reliable messaging and error handling

Transport, Routing and Packaging - 2

- Identify messaging routing
- Meet security requirements
- Provide for audit trails
- Define and meet acceptable levels of quality of service
- Support platform independent interoperability
- Support restart and recovery
- Convergence with SOAP
  - Decision at Vancouver Plenary – February 2001

TRP Message Structure



Communication Protocol (SMTP, HTTP, etc.)

MIME multipart/related

SOAP-ENV:Envelope

SOAP-ENV:Header

eb:MessageHeader

eb:TraceRoute

ds:Signature

/SOAP-ENV:Header

SOAP-ENV:Body

eb:Manifest

eb:StatusData

SOAP-Env:Fault

eb:ErrorList

/SOAP-Env:Fault

eb:Acknowledgements

/SOAP-ENV:Body

/SOAP-ENV:Envelope

Payload

Payload

Business Process WG

- An opportunity to re-engineer using Internet technologies
- Model “vertical” business processes using a common methodology
  - UML based
  - Reusable templates in repository
- Map models to XML for implementation
  - Generate partner profiles
  - Provide security profiles

Core Components WG

- Analysis and Discovery processes
  - Use to create a catalogue of existing components
  - Domain experts analyse their industry
  - Components assessed to eliminate overlap
- Propose methodology for core component development
  - Naming conventions
  - Contexts for re-usability
  - Application of assembly and context rules
- Components available via Registry/Repository

Registry/Repository WG

- Define an ebXML Registry and associated Information model which will allow:
    - Discovery of trading partners and their profiles
    - Discovery of business process capabilities and communications specifications
    - Discovery of data interchange specifications used within the context of a business process
    - Retrieval of software component adapters to integrate information into business applications
    - Development of business process models
    - Discovery of core business objects and core components

Trading Partner WG

- Rules of interaction between independent businesses
  - Independent of internal business processes
- An XML Document
- Partner Profile lists what a partner CAN do – their IT capabilities
  - Communication protocols
  - Security requirements
  - Business processes they support
- Partner Agreement lists what partners WILL do
  - An intersection of two partner profiles
  - With some negotiation
- Allows automatic generation of implementation at each party
  - Formal specification avoids misinterpretation
  - Assures each party configured compatibly

Security WG

- Provide a consolidated security model across ebXML
  - Policies
    - Privacy, non-repudiation, audit, ...
  - Technology
    - S/MIME DSIG, XML DSIG, S/MIME encryption, https .....
- Define security responsibilities for ebXML WGs
  - TP define security elements for partner profiles
  - BP/CC express security roles and policies
  - TRP support security technologies in partner profile
  - RR provide appropriate policies for access .....

Proof of Concept WG

- Assess the viability of ebXML specifications by demonstration and implementation
- “Marketing” role for ebXML
- Feedback to technical WGs
  - What does and doesn’t work
  - Specification refinement

Other WGs

- Technical Architecture
- Marketing and Awareness
- Quality Review

ebXML Summary

- ebXML successfully generating XML based specifications for e-business
- Migrating EDI community to XML
- Completion by May 2001
- ebXML will continue to a second phase
  - As of February 2001, the format has not been finalised

Links

- ebXML
  http://www.ebxml.org
- OASIS
  http://www.oasis-open.org
- UN/CEFACT
  http://www.unece.org/cefact
- XML.org
  http://www.xml.org

UDDI and Web Services

What is UDDI ?

- Universal Description, Discovery, and Integration
- A project to speed interoperability and adoption for web services
  - Standards-based specifications for service description and discovery
  - Shared operation of a web-based business registry
  - Partnership among industry and business leaders - more than 200 companies have signed up so far

What problems does UDDI solve ?

Broader B2B

A mid-sized manufacturer needs to create 400 online relationships with customers, each with their own set of standard and protocols

Describe Services

Smarter Search

A flower shop in Australia wants to be “plugged in” to every marketplace in the world, but doesn’t know how

Discover Services

Easier Aggregation

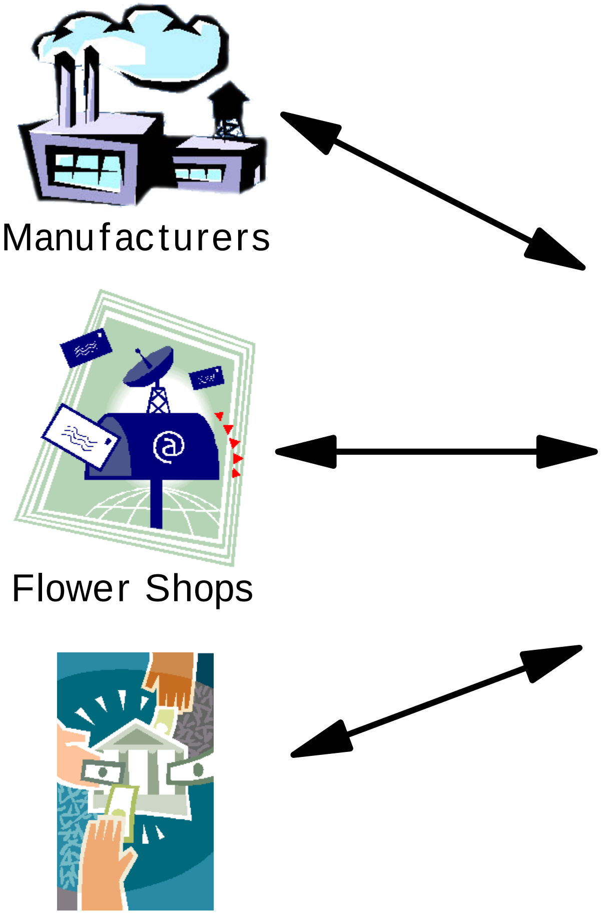
A B2B marketplace cannot get catalog data for relevant suppliers in its industry, along with connections to shippers, insurers, etc.

Integrate them Together

UDDI Vision and Process

1. Start with existing standards
   - TCP/IP, HTTP, XML
   - Industry-specific schemas
   - Shared vision of open protocols
2. Augment and implement via a Web Service
   - Common web services “stack”
   - Shared implementation to avoid confusing customers
   - Public specs, open service, inclusive process
3. Transition to a Standards Body
   - Manage design process for 3 revisions

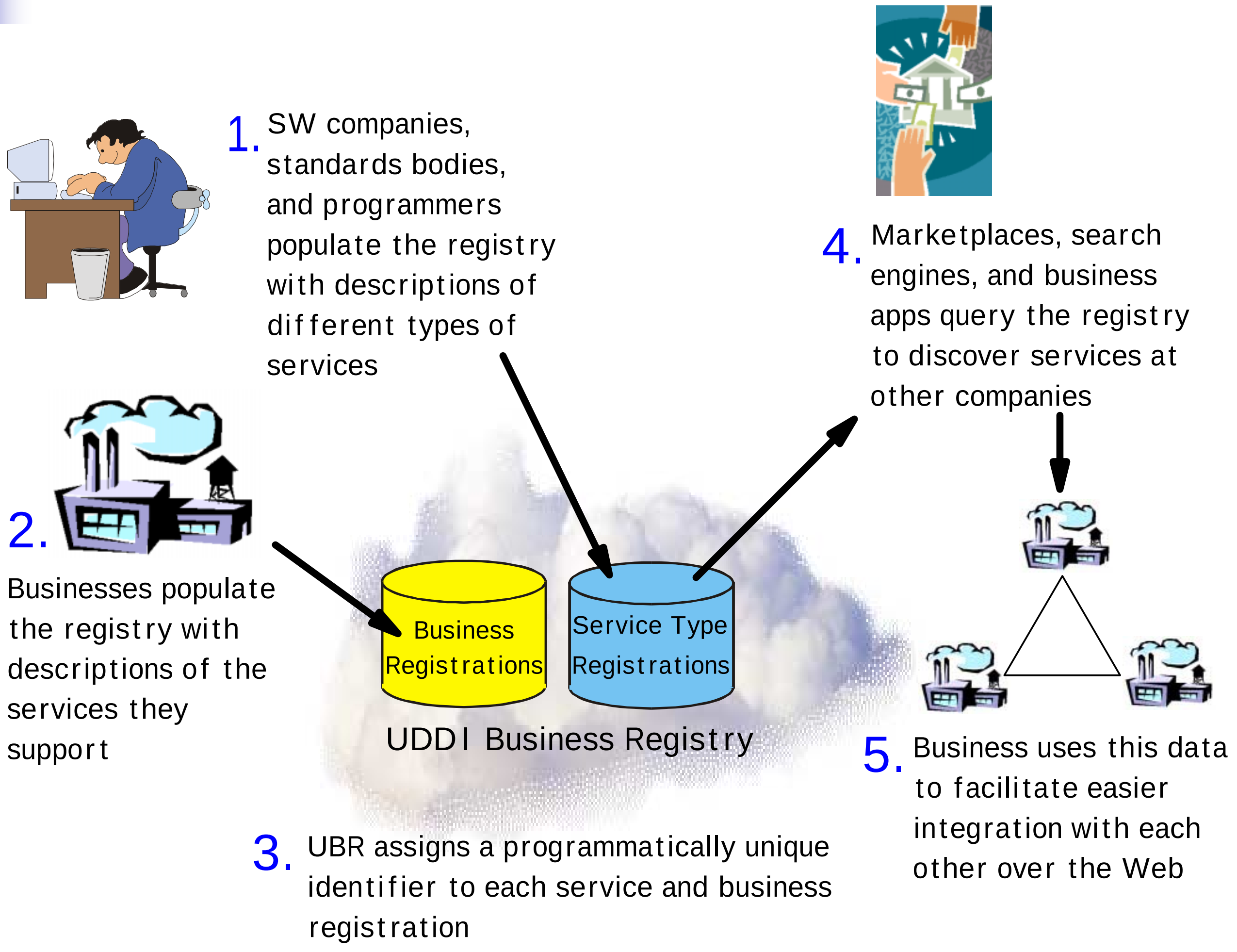
UDDI v1 Implementation



UDDI Business Registry

- Programmatic descriptions of web services
- Programmatic descriptions of businesses and the services they support
- Programming model, schema, and platform agnostic
- Uses XML, HTTP, and SOAP
- Free on the Internet

How UDDI v1 works



1. SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services

2. Businesses populate the registry with descriptions of the services they support

Business Registrations

Service Type Registrations

UDDI Business Registry

3. UBR assigns a programmatically unique identifier to each service and business registration

4. Marketplaces, search engines, and business apps query the registry to discover services at other companies

5. Business uses this data to facilitate easier integration with each other over the Web

And now …..

Let’s look at Web Services

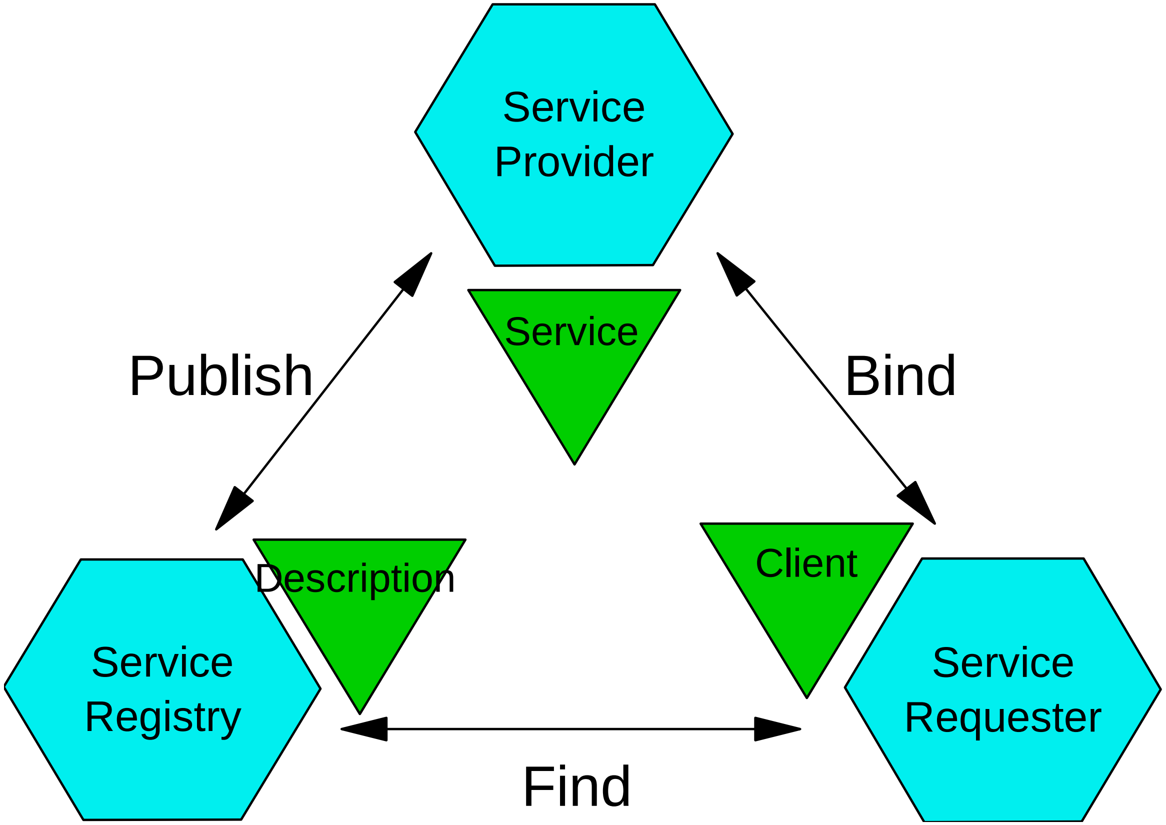
in more detail

Defining Web Services

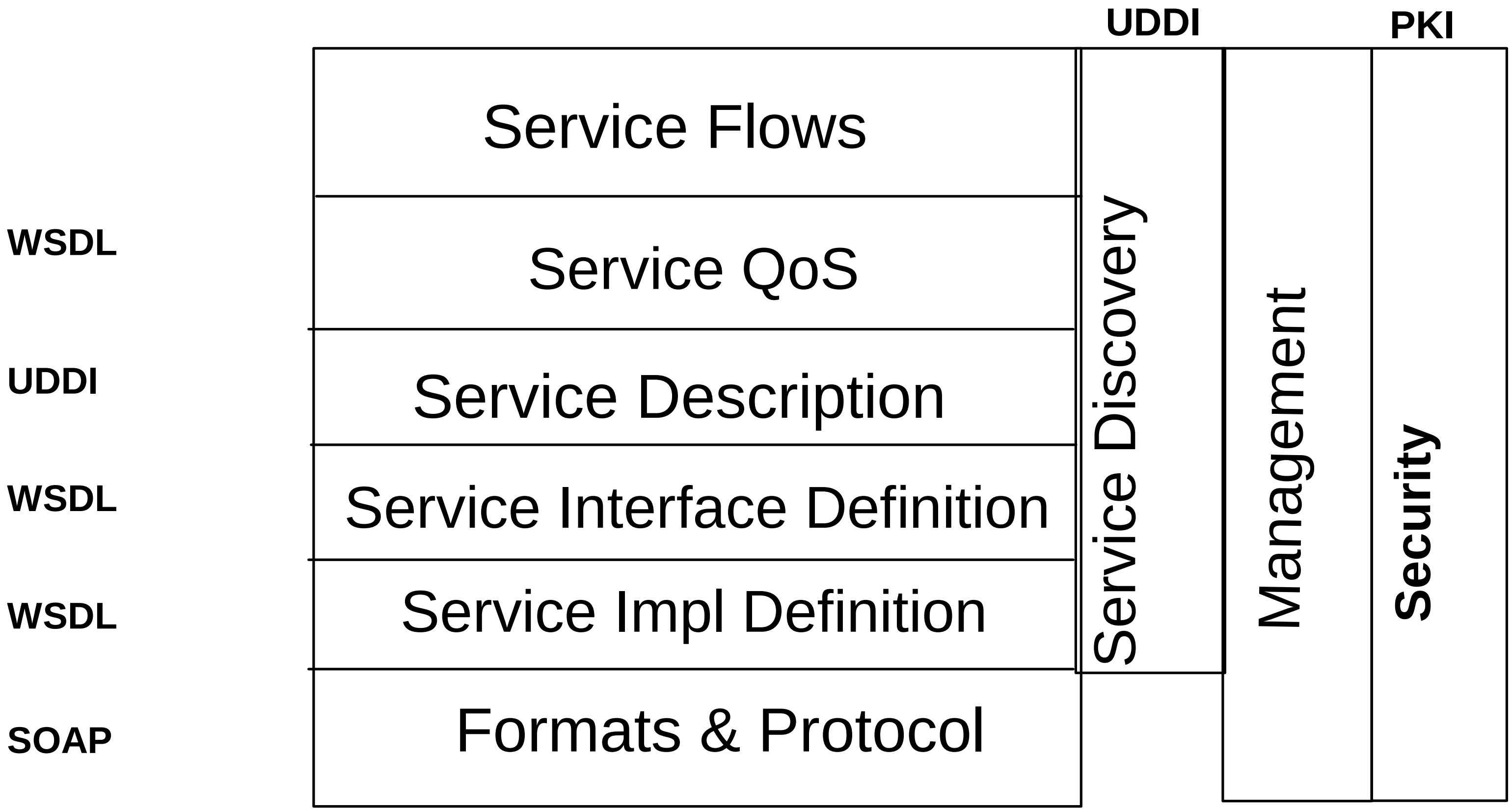
Modular applications that can be:

- **Described** using a service description language. WSDL (Web Services Description Language).
- **Published** by registering its description and use policies with a registry.
- **Found** by sending queries to that registry and receiving the binding details of the service(s) that fit the parameters of the query.
- **Bound** by using the information contained in the service description to create a callable service instance or proxy.
- **Invoked** over a network by using the information contained in the binding details of the service description.
- **Composed** with other services into new services.

Conceptual Foundation



Web Services Stack



UDDI

PKI

Service Flows

WSDL

Service QoS

UDDI

Service Description

WSDL

Service Interface Definition

WSDL

Service Impl Definition

SOAP

Formats & Protocol

Service Discovery

Management

Security

Definition of Stack Layers

- Formats and Protocols
  - Messages are sent as XML documents conformant to well known or published XML Schemas.
  - The messaging infrastructure will use W3C XML Protocol (SOAP until W3C XP available) for the message envelope and common standard.

Definition of Stack Layers

- Service Interface Definition
  - Specification of logical interface
  - WSDL provides all IDL capabilities
- Service Implementation Definition
  - Defines network location, protocols, security requirements and other attributes specific to a particular instance of a service
  - WSDL provides this as well

Definition of Stack Layers

- Service Description
  - the nonfunctional service description
  - Taxonomy, ownership, business name, business type, and various keywords that make the discovery easier.
  - UDDI provides this capability

Definition of Stack Layers

- Service Discovery
  - Dynamic at runtime
  - Static – services found and bound to at development time
    - Tools browse/search
  - UDDI registry is a services directory
    - Contains service definitions
    - Programatically searchable

Definition of Stack Layers

- Quality of Service
  - Implementation level
    - E.g. Transactional, secure
  - Interface level
    - E.g. Tolerable response times
  - Provided by WSDL and potentially other definitions
    - It is possible to extend WSDL with concepts based on ebXML partner agreements

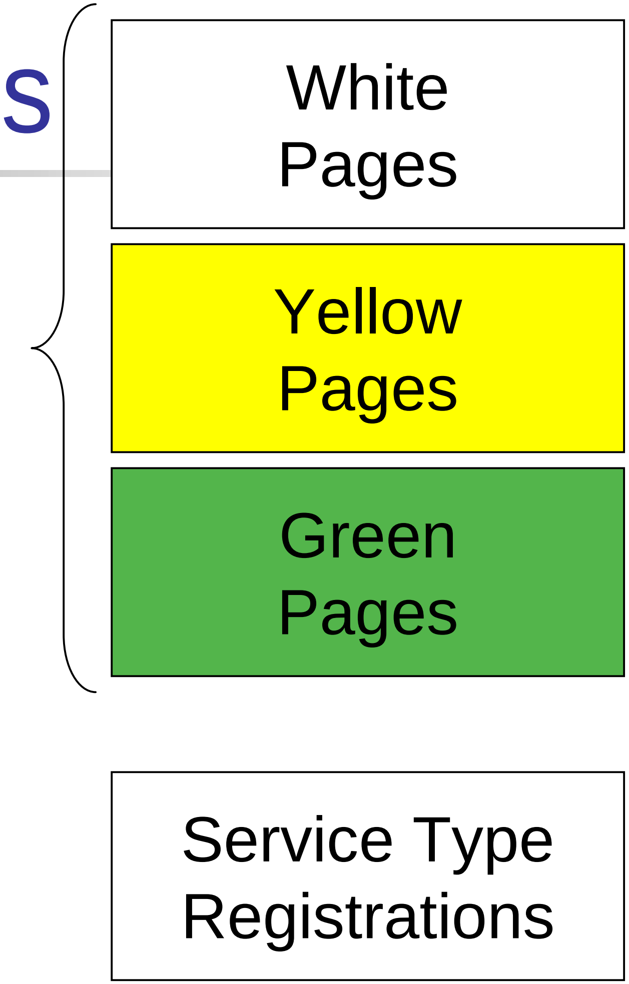
Definition of Stack Layers

- Service Flows
  - Services as activities in work flows
  - Flows of services, potentially between partners
  - Services as wrappers for exposed business processes
  - Service composition
  - This is an area of study

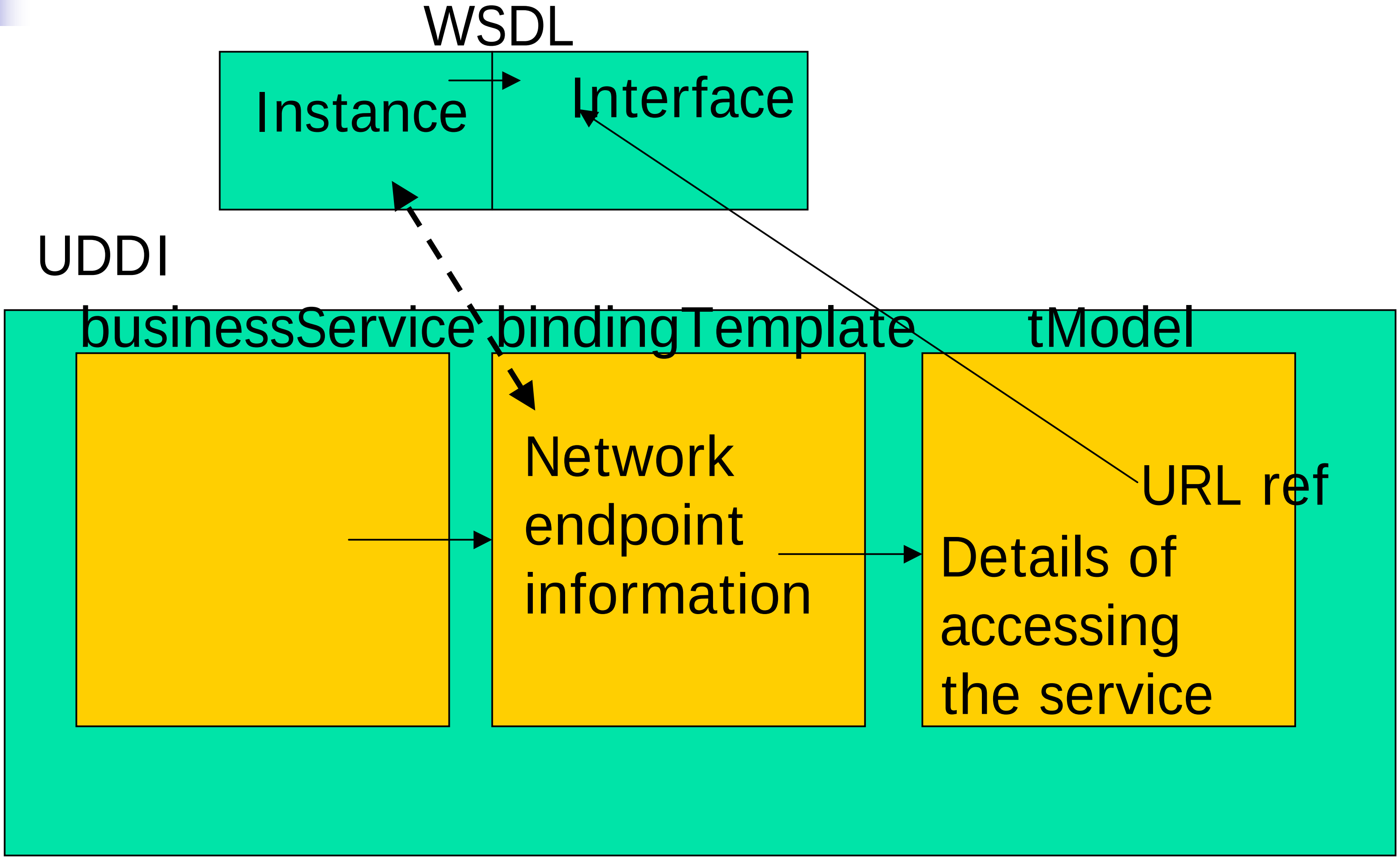
Registry Contents

Businesses register
public information
about themselves

Standards bodies, programmers,
businesses register information
about their Service Types



UDDI, WSDL Relationships



WSDL

Instance

Interface

UDDI

businessService

bindingTemplate

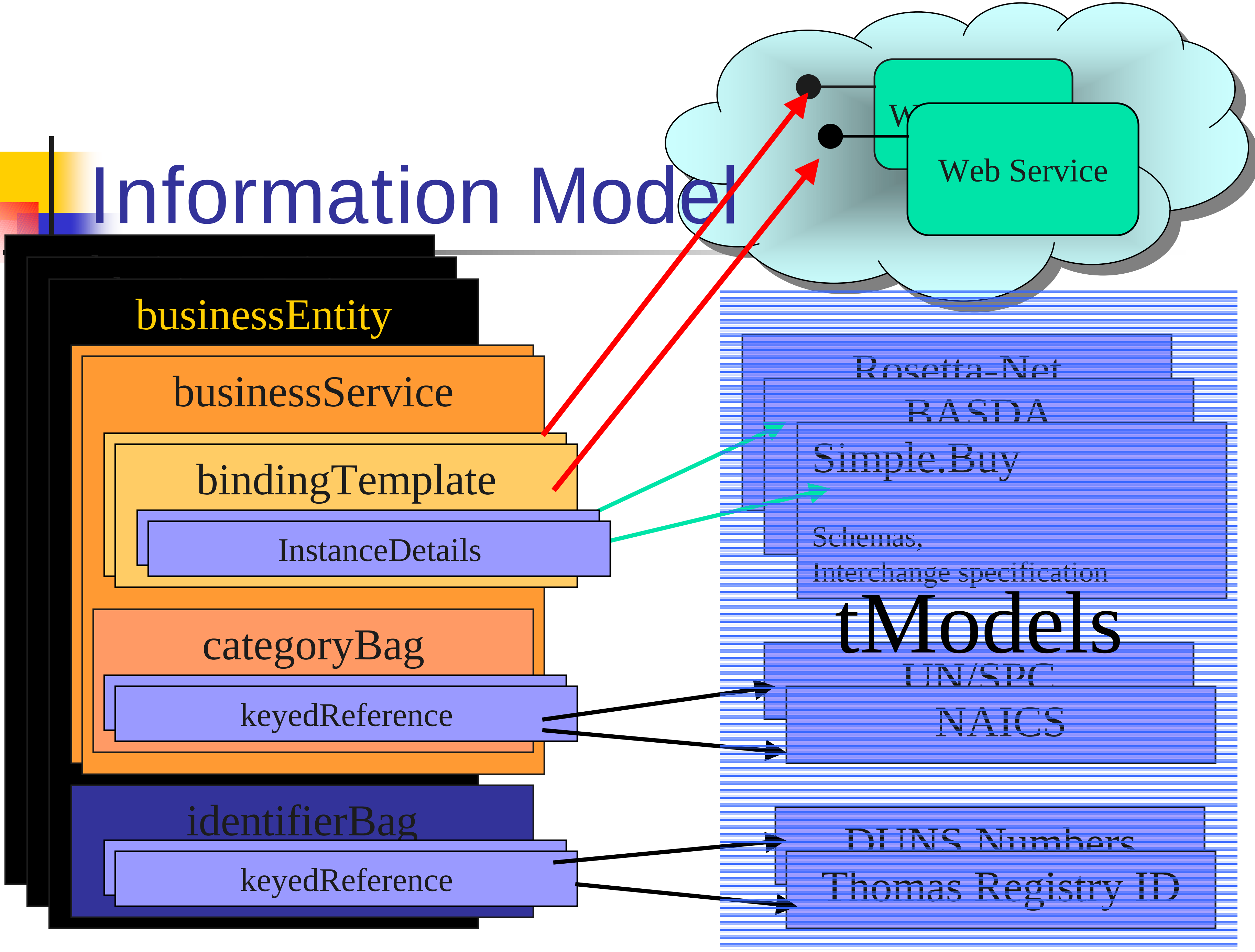
tModel

Network endpoint information

URL ref

Details of accessing the service

Information Model



W

Web Service

businessEntity

businessService

bindingTemplate

InstanceDetails

categoryBag

keyedReference

identifierBag

keyedReference

Rosetta-Net

BASDA

Simple.Buy

Schemas,
Interchange specification

tModels

UN/SPC

NAICS

DUNS Numbers

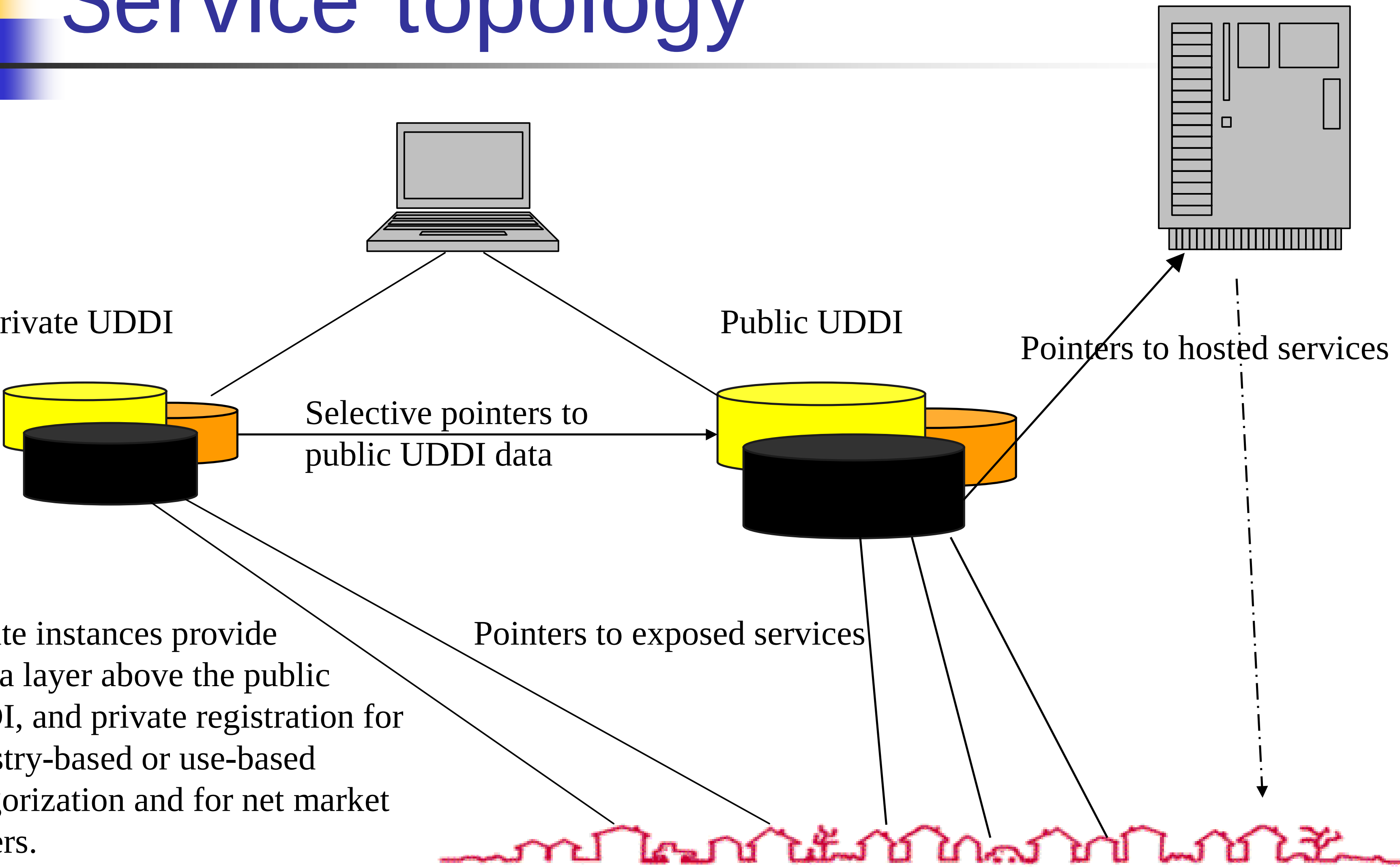
Thomas Registry ID

From 500 Feet – Service invocation

- Publishers interface
  - Save things
    - save_business
    - save_service
    - save_binding
    - save_tModel
  - Delete things
    - delete_business
    - delete_service
    - delete_binding
    - delete_tModel
  - security…
    - get_authToken
    - discard_authToken

- Inquiry interface
  - Find things
    - find_business
    - find_service
    - find_binding
    - find_tModel
  - Get details
    - get_businessDetail
    - get_serviceDetail
    - get_bindingDetail
    - get_tModelDetail
- Taxonomy interface
    - validate_categorization

Service topology


Private UDDI
Public UDDI
Pointers to hosted services
Selective pointers to public UDDI data
Pointers to exposed services

Private instances provide both a layer above the public UDDI, and private registration for industry-based or use-based categorization and for net market makers.

WSDL references in UDDI

```
<bindingTemplate>
         (...)
         <accessPoint urlType="http">
            http://example.com/stockquote
         </accessPoint>
         <tModelnstanceDetails>
            <tModelnstanceInfo>
               <tModelKey> (...) </tModelKey>
               <overviewDoc>
                  http://example.com/stockquote/stockquote.wsdl
               </overviewDoc>
               <instanceParms>
                  <port name="StockQuotePort"
                        binding="StockQuoteBinding"/>
               </instanceParms>
            </tModelnstanceInfo>
         <tModelnstanceDetails>
      </bindingTemplate>
```

What uses UDDI?

- Tool building client (Service Consumer)
  - Browse or search registry
  - Create a service proxy
- Tool publishing the service
  - Generates WSDL
  - Construct UDDI entries
- Application that needs dynamic binding
  - Directly access UDDI
  - Query can be pre-generated

How is UDDI Accessed?

- UDDI Spec defines the interfaces in terms of XML messages
- UDDI4J provides Java language mappings

```
UDDIProxy proxy = new UDDIProxy();
proxy.setInquiryURL("http://www-3.ibm.com/services/uddi/ testregistry/inquiryapi");
proxy.setPublishURL("https://www-3.ibm.com/services/uddi/
    testregistry/protect/publishapi");
```

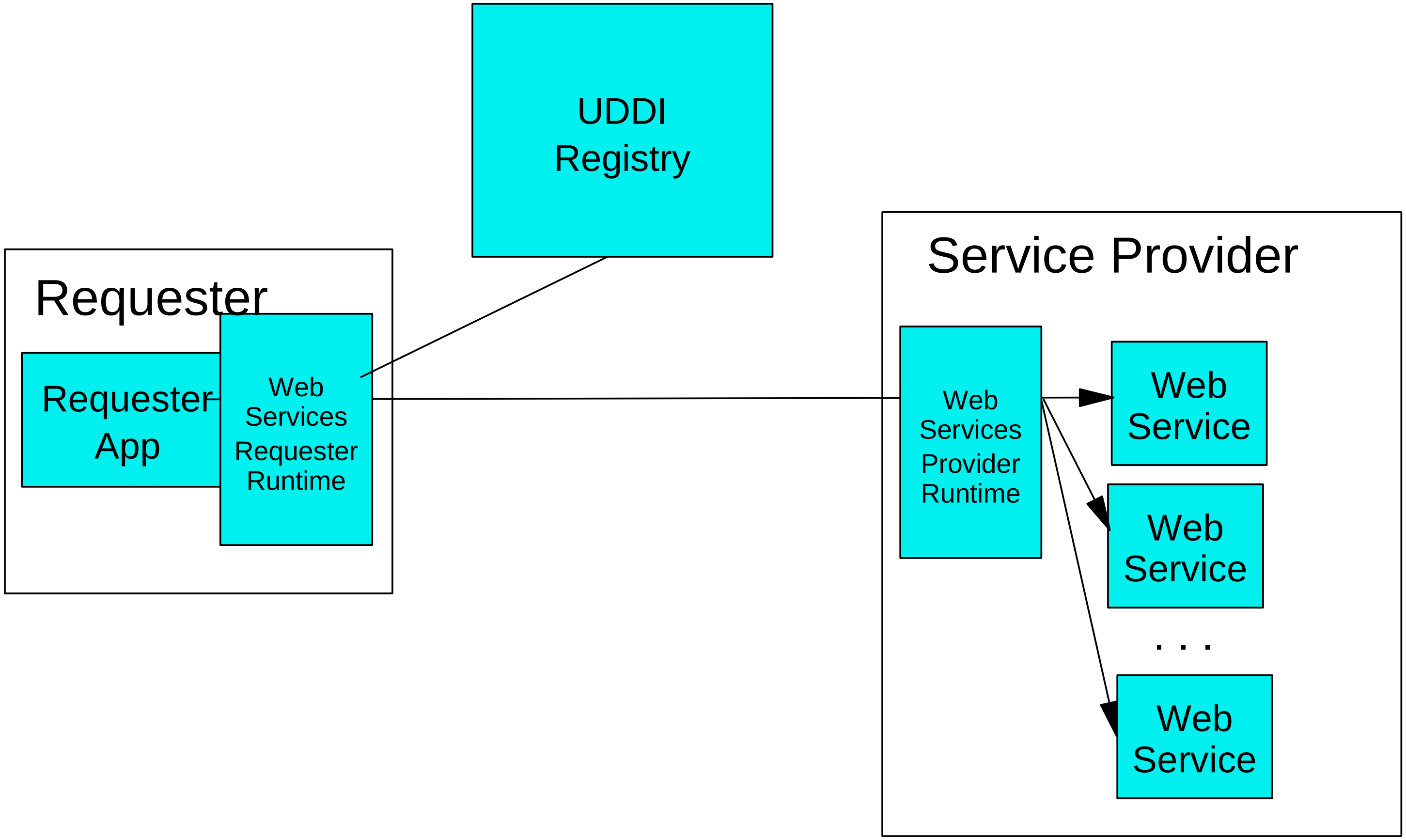
Publishing a WSDL-defined Service

- WSDL instance information (implementation: ports, addresses, etc) converted to UDDI bindingTemplate elements.
- URL of WSDL interface definition is stored in overviewURL element of UDDI tModel.

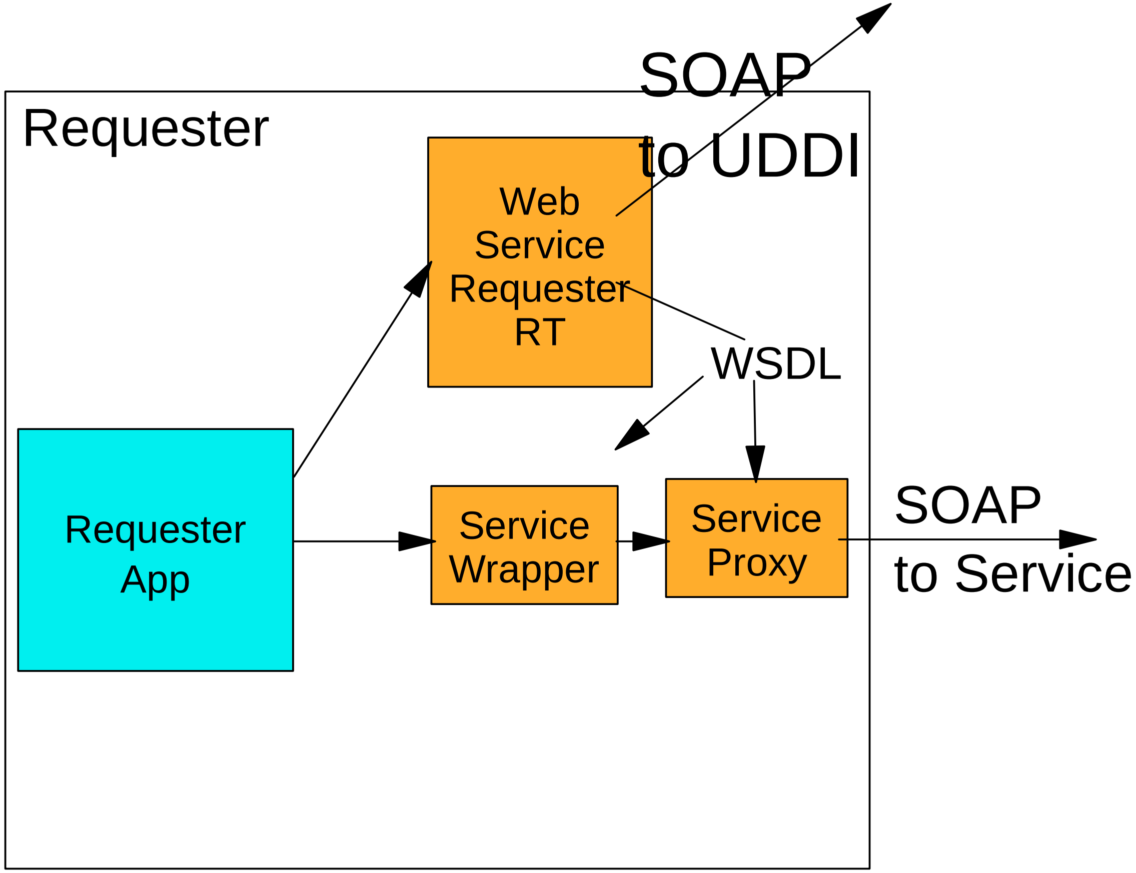
Binding to a Service

- Done by Web Services runtime/tools
  - Create needed WSDL instance information from bindingTemplate
  - Obtain interface WSDL reference from the tModel
  - Construct service proxy

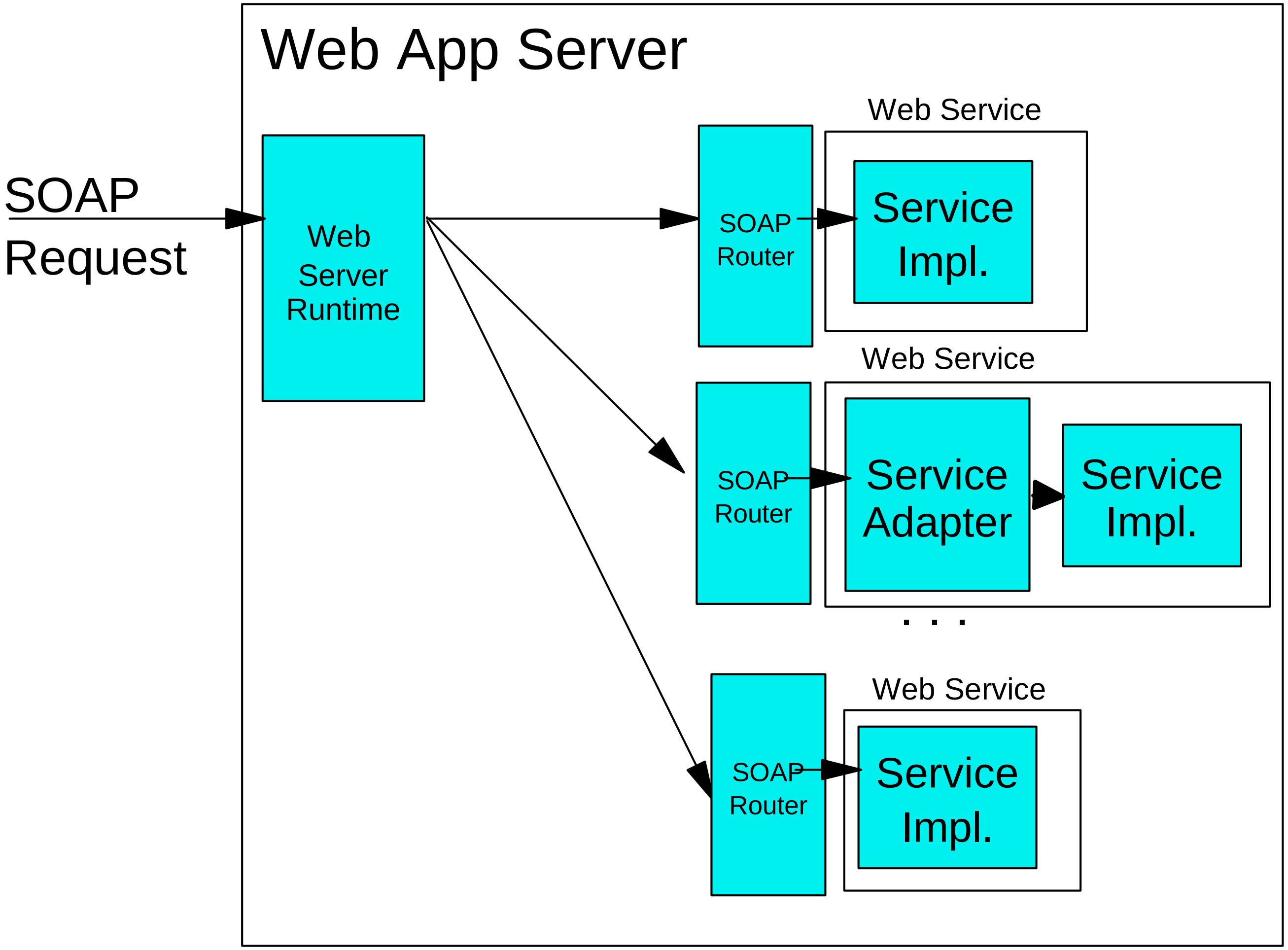
Architectural Components



Service Requester Components



Provider Components



SOAP Request

Web App Server

Web Server Runtime

Web Service

SOAP Router

Service Impl.

Web Service

SOAP Router

Service Adapter

Service Impl.

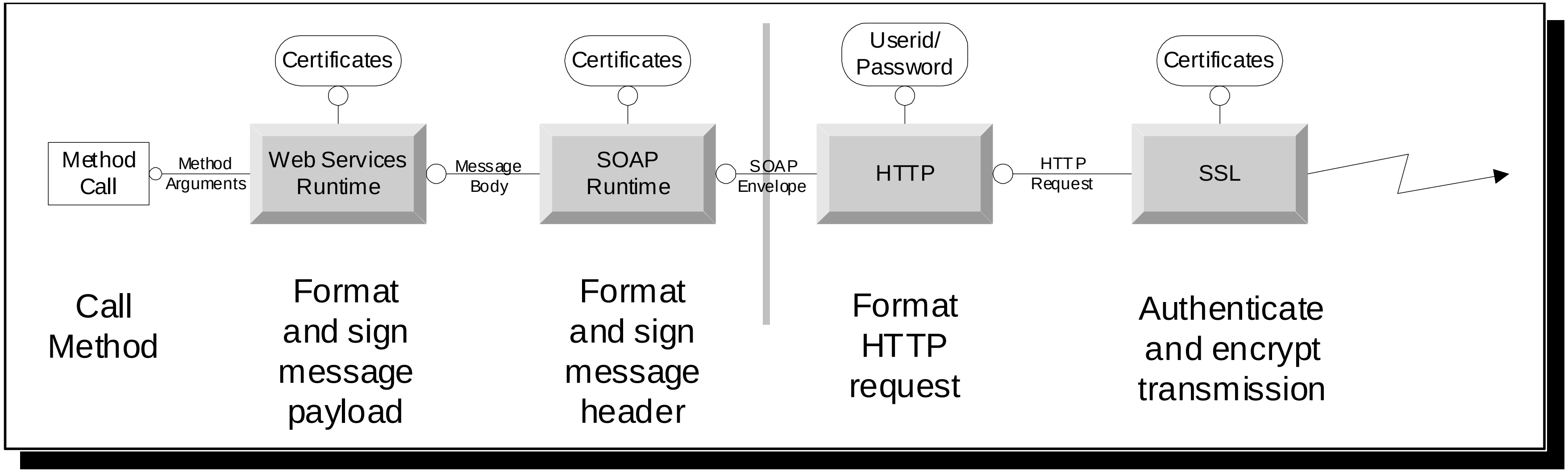
. . .

Web Service

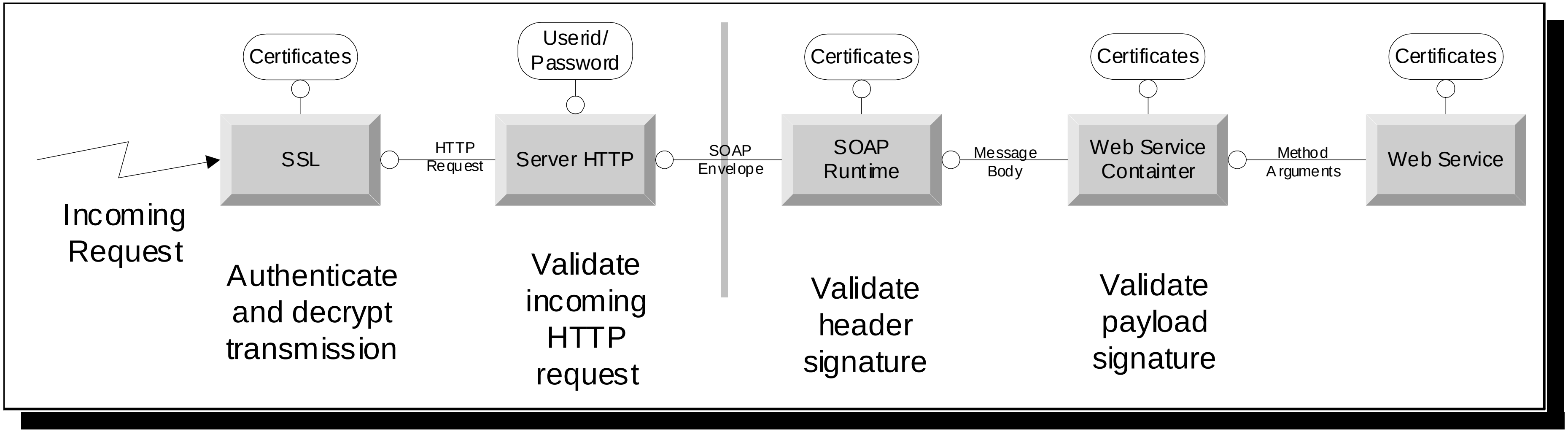
SOAP Router

Service Impl.

Security Scenario - Requester



Security Scenario – Provider



Incoming Request

Certificates

SSL

Authenticate and decrypt transmission

HTTP Request

Userid/ Password

Server HTTP

Validate incoming HTTP request

SOAP Envelope

Certificates

SOAP Runtime

Validate header signature

Message Body

Certificates

Web Service Containter

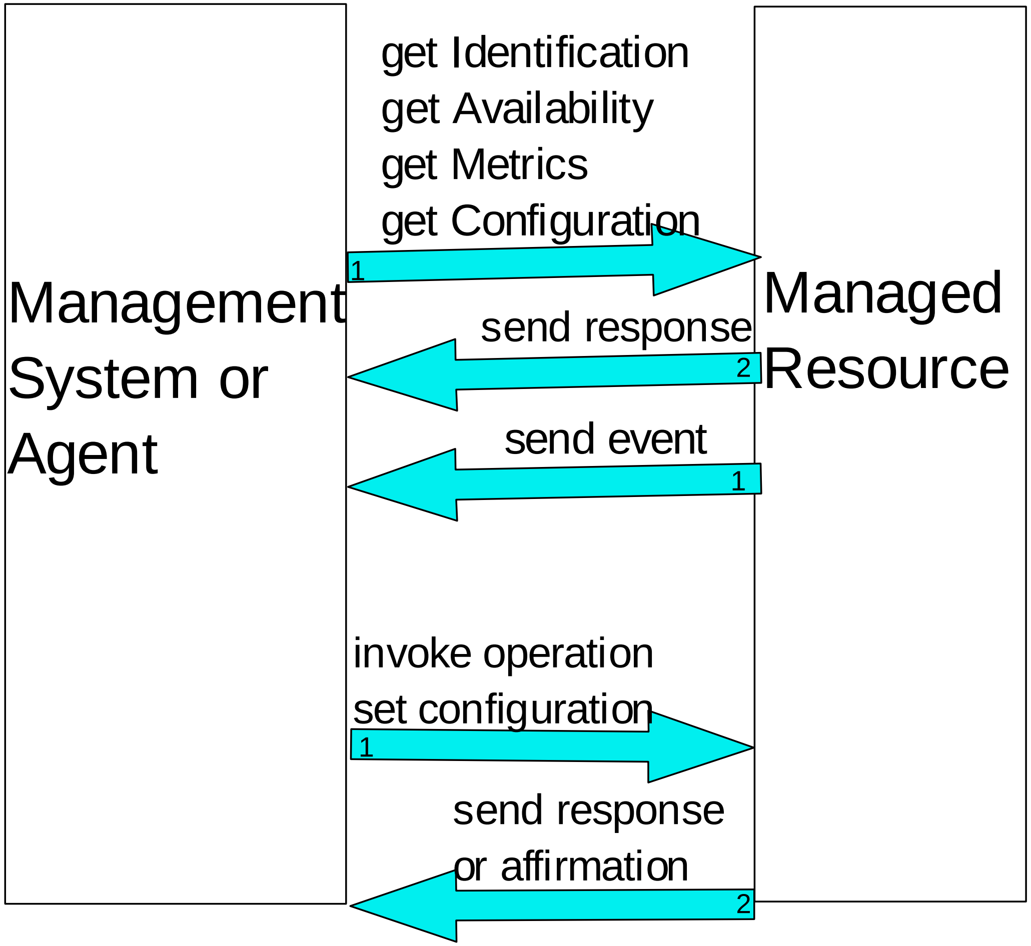
Validate payload signature

Method Arguments

Certificates

Web Service

Management of Web Services - Basic Concepts



Management of Web Services - specifics

Example of IBM Web Services Toolkit

- Based on Java Management Extensions
- Metrics, availability and identification,etc., handled with out service action
  - Management Beans automatically created
- Advanced management functions
  - Configuration
    - Availability requires explicit support by the service implementation

Application Development Scenario - 1

Create new Web Service implementing an existing interface

1. Use the UDDI browser to find the service and download WSDL documents
2. Assuming the SOAP protocol, use the SOAP tools to create a corresponding SOAP deployment descriptor.
3. Create the classes that implement the service described in the WSDL documents
4. Use the UDDI editor to create the additional UDDI information
5. Publish service information to a test UDDI server and test
6. Promote service to a production UDDI server

Application Development Scenario - 2

Create new Web Service implementing a new interface

1. Create an Interface
2. Create implementation classes.
3. Use the WSDL generator to derive the WSDL Interface document from the interface in step 1. Assuming the SOAP protocol, use the SOAP tools to create a corresponding SOAP deployment descriptor.
4. Use a WSDL editor to complete the WSDL implementation document.
5. Use a UDDI editor to create the additional UDDI information.
6. Publish the service information to a test UDDI server and test. Promote the service information to a production UDDI server.

Application Development Scenario - 3

Expose an existing application as a Web Service

1. Use a WSDL generator to derive the WSDL interface document from an existing application
2. Use a service adapter generator to create the web service adapter
3. Assuming SOAP, use tools to create a corresponding SOAP deployment descriptor
4. Publish the service information to a test UDDI server and test Promote the service information to a production UDDI server

Service Consumer

Using Service wrapper, generated by tooling, invoke the service as a local object.

```
WeatherService service = new WeatherService();

        String weather = null;
        String[] locations = null;
        if (location!=null)
          weather = service.getWeather(location);
        if (state!=null)
           locations = service.getLocations(state);
```

WSDL Definition of a Service

- WSDL Document contains
  - Interface Definition
    - Name
    - Input message
    - Result message
  - Specifics of an implementation of that interface
    - Transport
    - Encoding
    - Location
  - Any XML types needed to express the interface

Interface Definition

```
<?xml version="1.0"?>
<definitions name="AddressService"
                  targetNamespace="urn:show-address“
                 xmlns="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/1999/XMLSchema”
                  xmlns:xsd1="http://www.addressbook.com/ns/ShowAddress"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

  <message name="AddressInput">
    <part name="theAddress" type="xsd1:address"/>
  </message>

  <portType name="AddressHandler">
    <operation name="printAddress">
      <input message="AddressInput"/>
    </operation>
  </portType>
```

Instance (Interface) Definition

```
<binding name="AddressSoapBinding" type="AddressHandler">
  <soap:binding style="rpc"
       transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="printAddress">
   <soap:operation soapAction=""/>
   <input>
    <soap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
             http://www.ibm.com/namespaces/xmi"
        namespace="urn:show-address"/>
   </input>
  </operation>
 </binding>
```

Instance (location) Definition

```
<service name="AddressService">
    <port name="AddressPort" binding="AddressSoapBinding">
      <soap:address location="http://localhost:4040/soap/servlet/rpcrouter"/>
    </port>
  </service>
```

Reach via SOAP at this location

Data Type Definition

```
<types>
    <xsd:schema targetNamespace="http://www.addressbook.com/ns/ShowAddress"
            xmlns:xsd="http://www.w3.org/1999/XMLSchema">
      <xsd:complexType name="address">
        <xsd:element name="street" type="xsd:string"/>
        <xsd:element name="city" type="xsd:string"/>
        <xsd:element name="state" type="xsd:string"/>
        <xsd:element name="zip" type="xsd:string"/>
      </xsd:complexType>
    </xsd:schema>
  </types>

</definitions>
```

Referenced in Interface Definition

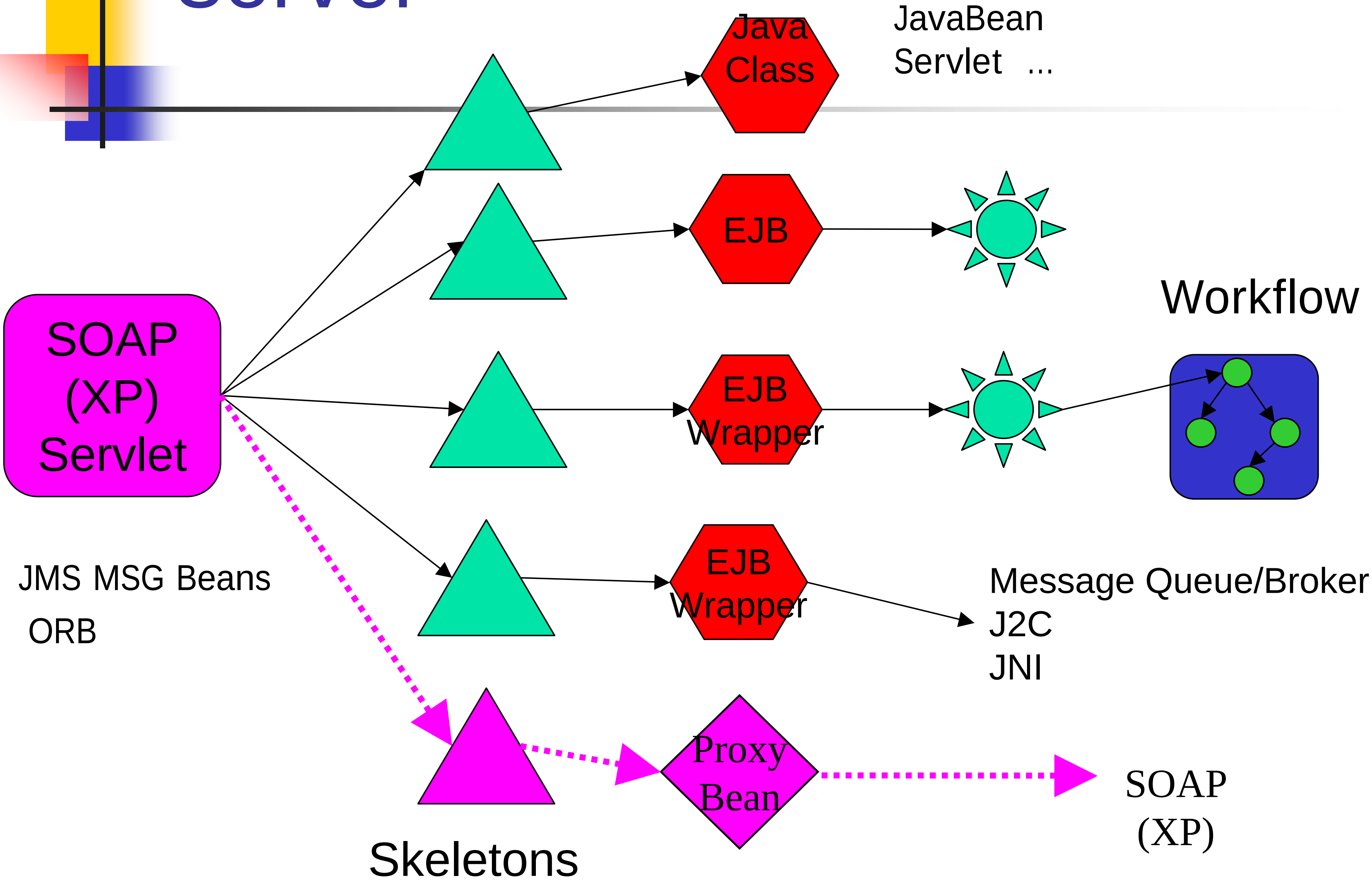
Implementation Example

```
import java.net.*;
import org.apache.soap.*;
import com.addressbook.www.ns.ShowAddress.*;
public class ShowAddressClient
{
  public static void main(String[] argv) throws MalformedURLException, SOAPException
  {
    if (argv.length != 4)
    {
      System.err.println("Usage:\n" + "  java " + ShowAddressClient.class.getName() +
                         " street city state zip");
      System.exit(1);
    }
    Address addr = new Address(argv[0], argv[1], argv[2], argv[3]);
    AddressHandlerProxy testProxy = new AddressHandlerProxy();
    testProxy.printAddress(addr);
  }
}
```

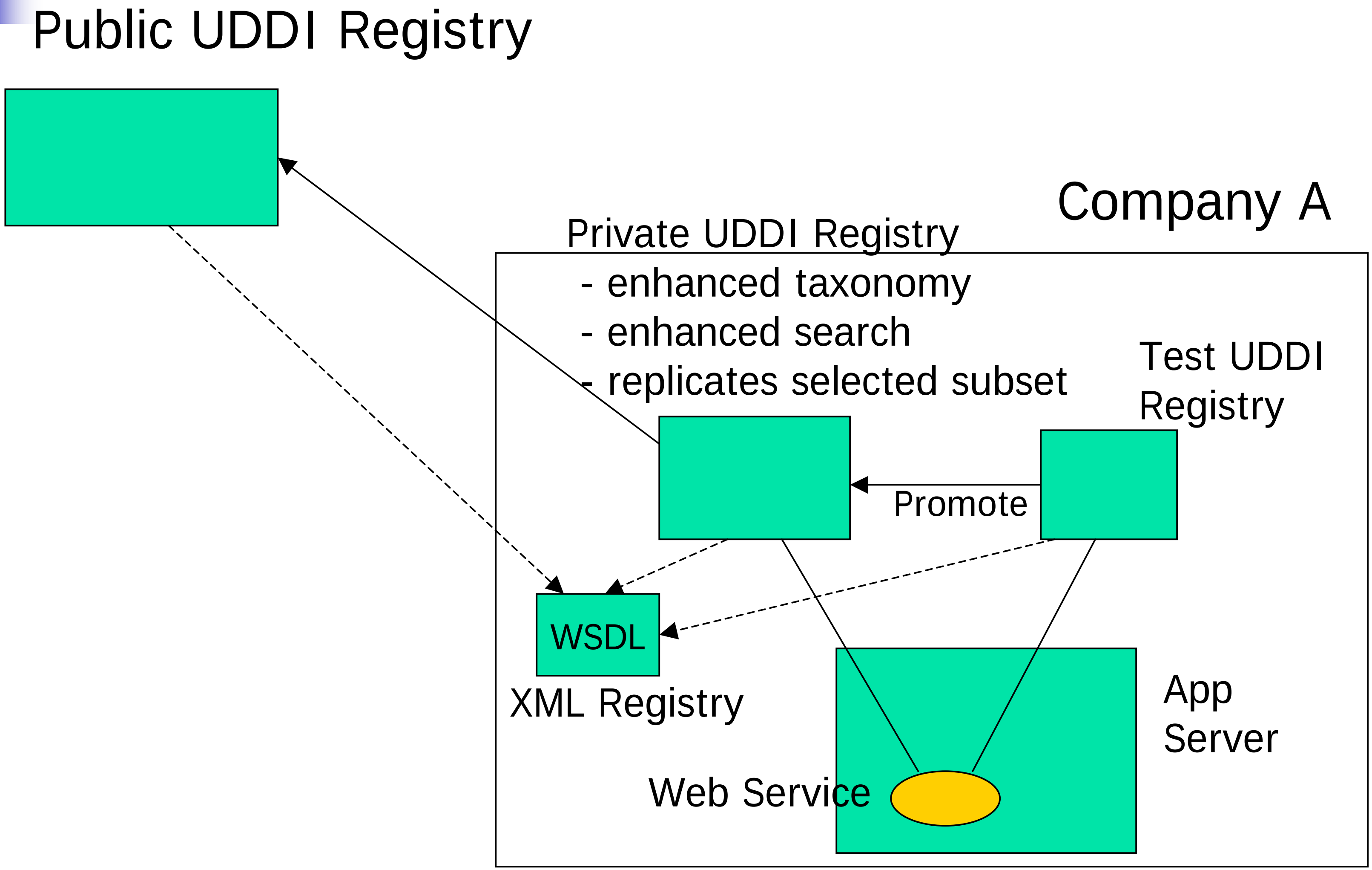
Tooling Support

- Example of Web Services Tooling is the IBM “XML and Web Services Development Environment”
  - Logical tool components mentioned in scenarios are provided by this package
  - Early version on alphaWorks now

Web Services in a Web App Server



Operational Topology



Public UDDI Registry

Company A

Private UDDI Registry
- enhanced taxonomy
- enhanced search
- replicates selected subset

Test UDDI Registry

Promote

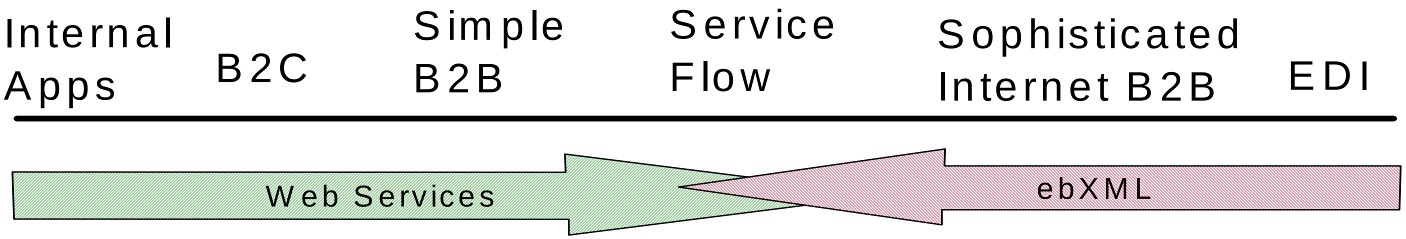
WSDL

XML Registry

App Server

Web Service

Relationship to ebXML



Technologies are complementary and will both be part of the evolution of the plumbing for doing business on the web.

For More Information

- UDDI

  uddi.org
- IBM developerWorks Web Services Zone

  www.ibm.com/webservices
- IBM alphaWorks

  www.alphaworks.ibm.com
- W3C XML Protocol

  www.w3.org

Summary - 1

- The web owes its existence and success to standards
- These must be open and robust
- Industry will build on core technologies
    - Built on W3C recommendations
    - Industry groupings such as UDDI and ebXML will take advantage of the technologies
- Open Source and Open Standards are complementary

Summary - 2

- We are moving from a web composed of documents to a web that also contains business services.
- Web services is a standards-based way to make applications discoverable and usable on the Internet (intranet)
- The industry is focusing on UDDI, SOAP, and WSDL as the foundations of the Web services approach
- Web services are gateways into enterprise applications

Questions .....

- Contacting me
  - John Ibbotson
  - Email: john_ibbotson@uk.ibm.com
  - Tel: +44 (0)1962 815188