PENNSTATE World Campus

SWENG568: Enterprise Integration

Lesson 4: Services and SOA Fundamentals

Introduction (1 of 11)

Learning Objectives

- Understand the concepts of services and (enterprise) service computing
- Master Web Services technical fundamentals
- Get familiar with XML, SOAP, WSDL, and UDDI

By the end of this week, make sure you have completed the readings and activities found in the Lesson 4 Course Schedule.

The Concepts of Services and (Enterprise) Service Computing (2 of 11)

The Concepts of Services and (Enterprise) Service Computing

Please read the review on Page 115-117 very carefully. The review is excellent for you to refresh your memory of what we have learned so far about different approaches to address integration challenges technically and historically in enterprises.

As we discussed in Lesson 1, when a software support written in FORTRAN, C, or the like that needed to be repeated was separated out as a simple procedure, such as a method, function, or subroutine, the most rudimentary form of software service was created. As program languages evolve (Figure 4.1) and computing technologies advance, software development pays much more attention to data/functionality reuse and robustness across business domains for improved business competiveness. As we discussed in last lesson, enterprise integration relies on common shared infrastructures to improve the reliability and scalability of integrated IT systems (Figure 4.2).

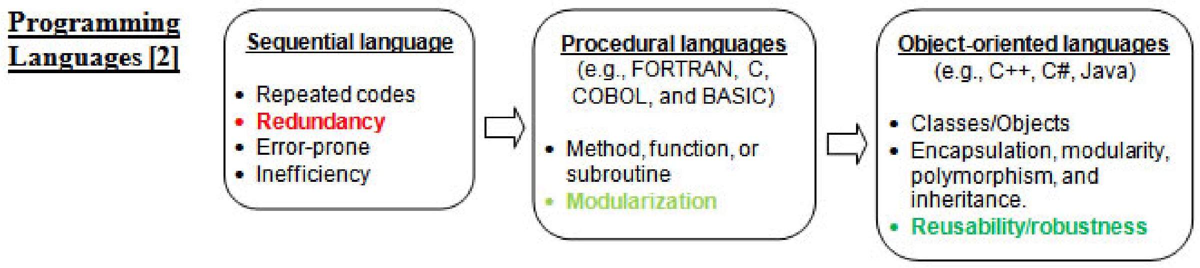
Programming Languages [2]
Sequential language
Repeated codes
Redundancy
Error-prone
Inefficiency
Procedural languages (e.g., FORTRAN, C, COBOL, and BASIC)
Method, function, or subroutine
Modularization
Object-oriented languages (e.g., C++, C#, Java)
Classes/Objects
Encapsulation, modularity, polymorphism, and inheritance.
Reusability/robustness

Figure 4.1: The Evolution of Programming Languages

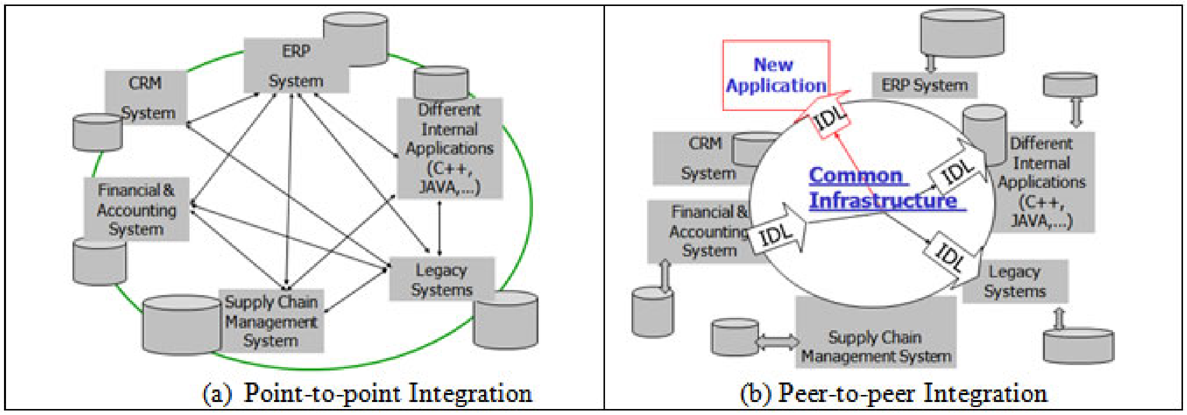
CRM System
ERP System
Different Internal Applications (C++, JAVA,...)
Financial & Accounting System
Legacy Systems
Supply Chain Management System
(a) Point-to-point Integration
New Application
ERP System
CRM System
Common Infrastructure
IDL
Different Internal Applications (C++, JAVA,...)
Financial & Accounting System
Legacy Systems
Supply Chain Management System
(b) Peer-to-peer Integration

Figure 4.2: The Evolution of Integration Architecture

Compared to point-to-point architecture, we understand that the improvement using peer-to-peer architecture (Figure 4.2(b)) will be significant. For better reliability and scalability, the common shared infrastructure should support asynchronous messaging. However, as various kinds of technological heterogeneity can exist in a large enterprise, to ensure applications integrated in the enterprise or in an inter-enterprise setting to be scalable, adaptable, and reliable becomes more challenges.

In enterprise integration, technological heterogeneity that largely exists in a large enterprise or in an inter-enterprise setting includes the following:

- **Middleware heterogeneity** – very often more than one type of middleware is used in a large enterprise as a different business unit might have deployed a different application to address its unit's specific needs. It is also typically true different units invested on their IT systems at a different time and different applications accordingly might provide different connectivity for integration as technologies evolve as time goes.
- **Protocol heterogeneity** – different communication protocols are used in different applications. When they want to communicate with each other, protocol mismatch can arise.
- **Synchrony heterogeneity** – applications use different communication modes, synchronous or asynchronous, to interact with each other to meet different needs of data/functionality sharing.
- **Diversity of data formats** – applications developed by different solution providers typically adopt different data formats as domain-specific data representations are pervasive. When integration across business domains requires data/functionality sharing, the data incompatibility issue arises.
- **Diversity of interface declarations** – different IDLs are used to declare software component interfaces.
- **No common place for service lookup** – different naming or interface registration services used by different applications.

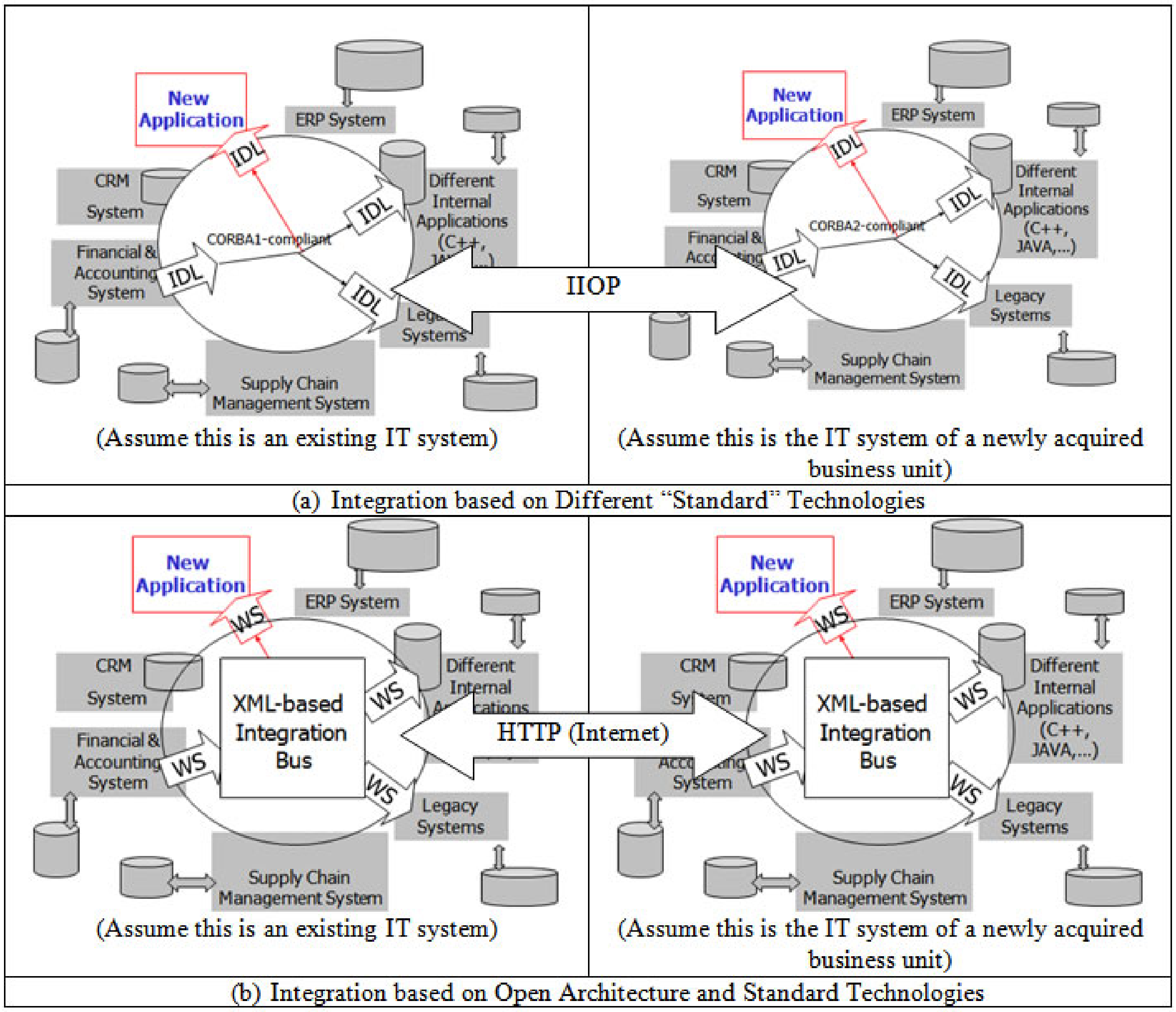
New Application
ERP System
CRM System
Financial & Accounting System
Different Internal Applications (C++, JAVA,...)
Legacy Systems
Supply Chain Management System
IDL
CORBA1-compliant
CORBA2-compliant
IIOP
(Assume this is an existing IT system)
(Assume this is the IT system of a newly acquired business unit)
(a) Integration based on Different "Standard" Technologies
WS
XML-based Integration Bus
HTTP (Internet)
(b) Integration based on Open Architecture and Standard Technologies

Figure 4.3: Evolution of Standard Solutions to Heterogeneous System

Given the existence of technological heterogeneity in enterprise integration, finding solutions to addressing the heterogeneity is necessary. As illustrated in Figure 4.2(b), in a given circumstance one kind of "standard" middleware can be used to assist the needed integration. However, there are many different kinds of middleware in industry. Even though an enterprise chose CORBA-compliant middleware technologies, integration would still get complicated when different kinds of CORBA middleware used in different facilities geographically dispersed across different regions (Figure 4.3(a)).

Apart from traditional software technologies, the Internet technologies (Figure 4.4) in general are **nonproprietary** and **platform-independent** although the Internet-based technologies continue to evolve and quite often be adopted and customized in a variety of computing environments. In this course, we focus on only the Web services. Using standard Internet protocols, the emerging Web services technology is a self-contained, self-describing, and network-neutral computing component. A Web service can be conveniently deployed, published, located, and invoked across the network. Web services can be assembled and reassembled as needed across the network, filling in the potential needs of adaptive enterprise computing in support of the needs of an integrated business (Figure 4.3(b)).

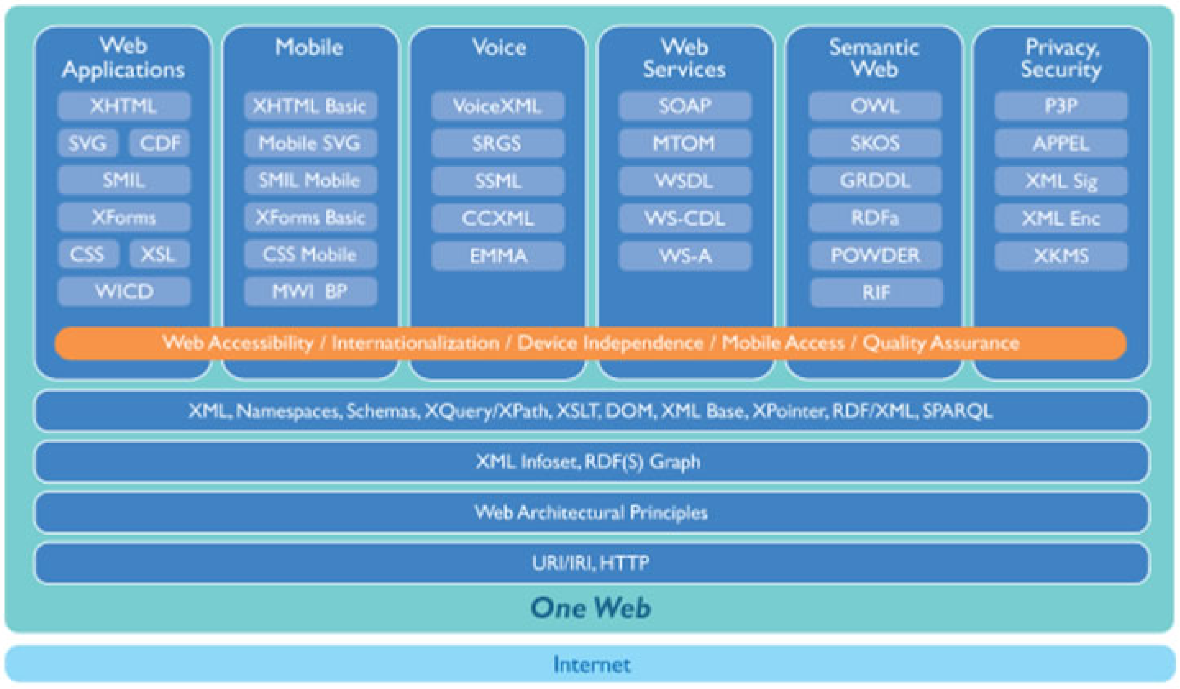

Figure 4.4: Internet Technology Stack Illustration (Source: W3C [6], Copyright © W3C

Whether data/functionality sharing at a micro- or macro-level (i.e., operational- or strategic-level) enabled by IT systems (i.e., integrated software systems) in an enterprise, to provide better data/information/knowledge services to assist or serve employees and customers, enterprises are eagerly embracing for building highly profitable service-oriented businesses through properly aligning business and technology and cost-effectively collaborating with their worldwide partners so that the best-of-breed services will be generated to meet the changing needs of customers. To be competitive in the long run it is critical for enterprises to be adaptive given the extreme dynamics and complexity of conducting businesses in today's global economy. In an adaptive enterprise, people, processes, and technology shall be organically integrated across the enterprise in an agile, flexible, and responsive fashion. Thus, the enterprise can quickly turn changes and challenges into new opportunities in this on demand business environment.

A generic service-oriented IT computing architecture for the development of a state-of-the-art enterprise network is illustrated in Figure 4.5. The top two layers represent services operations from the business process perspective while the bottom three layers shows the value-adding services processes from the computing perspective. Apparently, how to optimally align enterprise-level business strategies with value-adding operations/activities is the key to the success of the deployment of an agile enterprise service-oriented IT system. Web services are standard runtime technologies over the Internet, providing best ever mechanisms for addressing heterogeneous computing issues. By converging service-oriented architecture (SOA) and business process management (BPM) technologies, the concept of enterprise service computing emerges, representing the continuous evolution of the Web technology to support high performance, scalability, reliability, interoperability, and availability of distributed service-oriented IT systems in an intra- or inter-enterprise setting [3]. This lesson lays out the foundation for our next 3 lessons that will discuss more details on how the industry is heading to the open system architecture for adaptive enterprise service computing enablement.

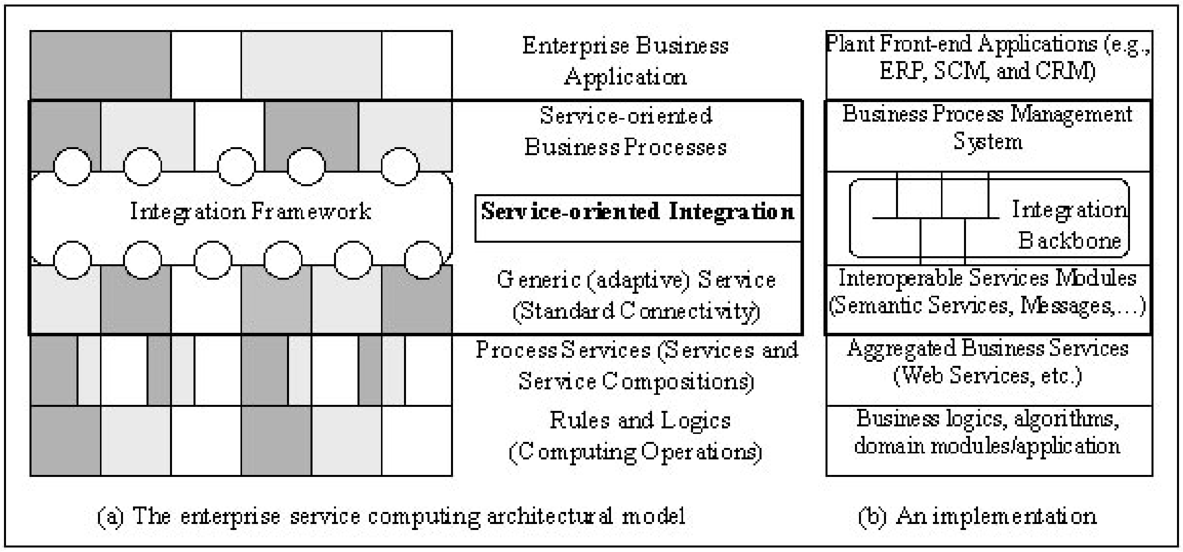
Enterprise Business Application
Service-oriented Business Processes
Integration Framework
Service-oriented Integration
Generic (adaptive) Service (Standard Connectivity)
Process Services (Services and Service Compositions)
Rules and Logics (Computing Operations)
(a) The enterprise service computing architectural model
Plant Front-end Applications (e.g., ERP, SCM, and CRM)
Business Process Management System
Integration Backbone
Interoperable Services Modules (Semantic Services, Messages,...)
Aggregated Business Services (Web Services, etc.)
Business logics, algorithms, domain modules/application
(b) An implementation

Figure 4.5: Service-Oriented Component Network Architectural Model (Adapted from [3]

Web Services Technical Fundamentals

The Web Services technology essentially consists of a stack of protocols and specifications for defining, creating, deploying, publishing, locating and invoking black network components (Figure 4.6). The stack primitively includes simple object access protocol (SOAP), XML and XML namespaces, web service description language (WSDL), and universal description, discovery and integration (UDDI). A computing service deployed as a Web service has to strictly comply with the stack of protocols and specifications. SOAP is the underlying communication protocol between the service provider and consumer, which explicitly defines how the service provider and consumer interact and what the enabled computation results. WSDL is the language for defining the computing service, which basically specifies the location of the computing service and the operations the service exposes. UDDI then provides the formal interface contract and the global base for the registration and discovery of the deployed computing service. Step-by-step, we will learn a great detail of the Web services in the following discussions.

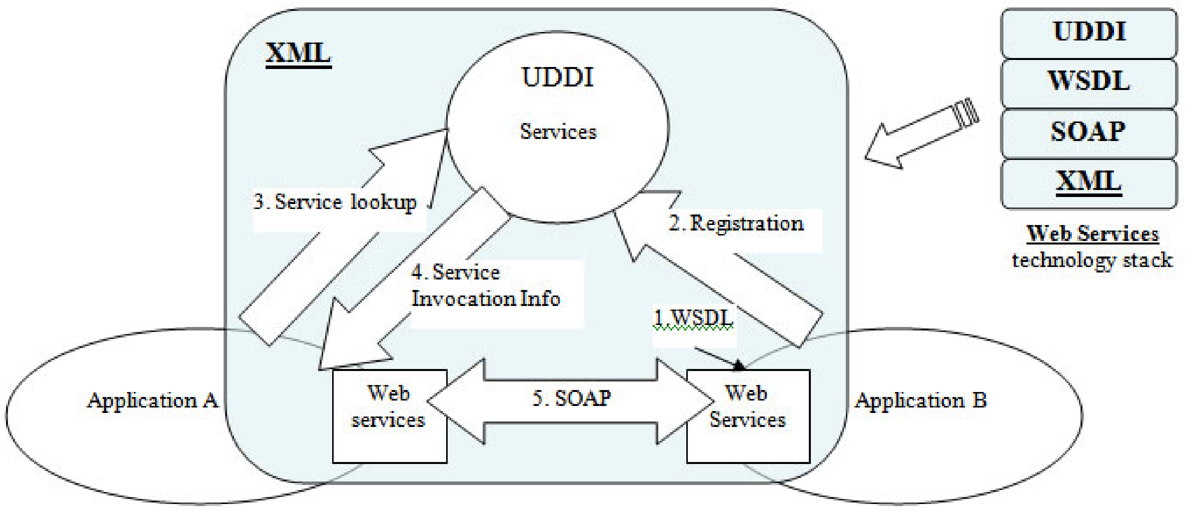

Figure 4.6 XML as the foundation of Web Service

XML Basics

What is XML [4, 5]?

- *XML is a standard markup language, standing for eXtensible Markup Language and recommended by W3C [4]*
- *XML tags are not predefined. Each user has to define its own or uses the tags defined in a given computing environment (e.g., complied with a DTD or schema)*
- *XML is written in a plaintext form and designed to be self-descriptive*
- *XML is used to carry data, not to display data*

Because XML is nonproprietary, programming language independent, and platform independent, XML becomes a "common ground" standard markup language for many web-based applications. Different from the situation not long ago that different community standards used in managing distributed objects and messages, XML gets accepted by all the communities nationally and internationally. XML is the foundation of Web services (Figure 4.6).

XML General Structure

“Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.”[7] XML is well-specified as a set of rules and conventions for designing text formats in a way that produces files or data that are easy to generate and read by any application [8]. The general structure of an XML document is illustrated in Figure 4.7, where element is repetitive. Listing 4.1 shows the data of 2 new students are described using an XML.

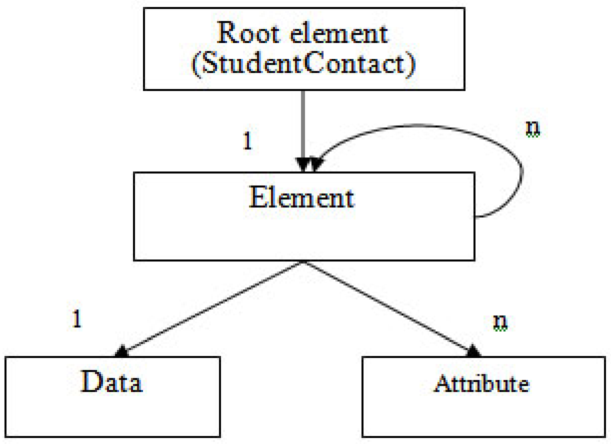

Figure 4.7 XML General Structure

Listing 4.1 Sample of XML Code for Student Contact Information (studentroster.xml)

```
<?xml version ”1.0” encoding = ”UTF-8”?>
<root>
    <row>
        <StudID>1111</StudID>
        <Name>Bob Smith</Name>
        <SSN>11l-222-3333</SSN>
        <EmailAddress>bsmith@psu.edu</EmailAddress>
        <HomePhone>215-222-2356</HomePhone>
        <HomeAddress>123 Tulip Road, Ambler, PA 19002</HomeAddress>
        <LocalAddress>321 College Ave, University Park, PA 16802</LocalAddress>
        <EmergencyContact>John Smith</EmergencyContact>
        <ProgramID>206</ProgramID>
        <PaymentID>1111-206</PaymentID>
        <AcademicStatus>1</AcademicStatus>
    </row>
    <row>
        <StudID>1112</StudID>
        <Name>Mary Thompson</Name>
        <SSN>132-566-1258</SSN>
        <EmailAddress> mthompson@psu.edu</EmailAddress>
        <HomePhone>212-234-4521</HomePhone>
        <HomeAddress>2300 Fifth Ave, NY, NY 10002</HomeAddress>
        <LocalAddress>328 College Ave., University Park, PA 16802</LocalAddress>
        <EmergencyContact>James Thompson</EmergencyContact>
        <ProgramID>207</ProgramID>
        <Payment1D>1112-207</Payment1D>
        <AcademicStatus>1</AcademicStatus>
    </row>
</root>
```

There are many XML editors available. Please try to edit some XML documents using an editor you like. If you haven’t done much XML programming at work, you can always try to use Microsoft Studio, XML Spy [9], oXygen XML Editor 11.1 (http://www.oxygenxml.com/index.html#new-version) [10], Open Source XML Editor [11], and many others, commercial or free.

XML Schema (6 of 11)

XML Schema

Each user could define his/her own XML structure. Although the defined XML structure is complied with XML specification, the use of different XML structure in terms of tags and data types will make application integration challenging. “An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself. These constraints are generally expressed using some combination of grammatical rules governing the order of elements, Boolean predicates that the content must satisfy, data types governing the content of elements and attributes, and more specialized rules such as uniqueness and referential integrity constraints.” [12]

Several different schema languages have been developed, Document Type Definitions (DTDs), Relax-NG, Schematron, and W3C XSD (XML Schema Definitions). W3C XSD is widespread used in enterprises around the world. An XML Schema essentially describes the structure of an XML document, which can be used [12]:

- to provide a list of elements and attributes in a given vocabulary;
- to associate types, such as integer, string, etc., or more specifically such as *StudID*, *Name*, *HomeAddress*, etc., with values found in documents;
- to constrain where elements and attributes can appear, and what can appear inside those elements;
- to provide documentation that is both human-readable and machine-processable;
- and to give a formal description of one or more documents.

A very simple example of an XML Schema is shown in Listing 4.2.

Listing 4.2 Sample of XML Schema for Student Contact Information (studentroster.xsd)

```
<?xml version="1.0" encoding”UTF-8”?>
<!—Document Root:root—>
<xsd:schema xmlns:xsd = "http://www.w3.org/200l/XMLSchema”>
    <xsd:element name = ”root”>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs=”O” name=”row” type=”rowType”/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name=”rowType”>
        <xsd:sequence>
            <xsd:element ref="StudID”/>
            <xsd:element ref="Name"/>
            <xsd:element ref=”SSN”/>
            <xsd:element ref=”Emai1Address”/>
            <xsd:element ref=”HomePhone”/>
            <xsd:element ref="HomeAddress”/>
            <xsd:element ref=”LocalAddress”/>
            <xsd:element ref=”EmergencyContact”/>
            <xsd:element ref=”ProgramID”/>
            <xsd:element ref="PaymentID"/>
            <xsd:element ref=”AcademicStatus”/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name=”StudID" type=”xsd:string”/>
    <xsd:element name=”Name” type=”xsd:string”/>
    <xsd:element name=”SSN" type=”xsd:string”/>
    <xsd:element name=”EmailAddress” type=”xsd:string”/>
    <xsd:element name=”HomePhone” type=”xsd:string”/>
    <xsd:element name=”HomeAddress” type=”xsd:string"/>
    <xsd:element name=”LocalAddress” type=”xsd:string"/>
    <xsd:element name=”EmergencyContact”type=”xsd:string”/>
    <xsd:element name=”ProgramID" type=”xsd:string”/>
    <xsd:element name=”PaymentID” type=”xsd:string”/>
    <xsd:element name=”AcademicStatus” type=”xsd:string”/>
</xsd:schema>
```

As discussed in [1], the namespace for a schema definition is http://www.w3.org/2001/XMLSchema (http://www.w3.org/2001/XMLSchema) as shown in Listing 4.2, which is linked to the prefix xsd. Of course, you can also use other prefix to indicate this namespace. The Schema definition defines many other parts that can be used in a Schema, such as element, attribute, simpleType/complexType (Listing 4.3), include, and import. Figure 4.8 shows the defined data types used in the XML Schema Definitions. XML Schema Tutorial [13] provides more detailed discussions on XML Schema.]

Listing 4.3 Sample of XML Schema and XML code snippet

```
<xsd:simpleType name=”productCode">
    <xsd:restriction base=”xsd:string">
        <xsd:pattern value=”d\{2)-d\{5 )"/>
    </xsd:restriction>
</xsd:simpleType>
Instance of productCode:
<productCode>12-12345</productCode>
<xsd:complexType name="telephoneNumber”>
    <xsd:sequence>
        <xsd:element name=”alea”>
        <xsd:simpleType>
                <xsd:restriction base=”xsd:string”>
            <xsd:pattern va1ue=”d\{3)"/>
        </xsd:restriction>
    </xsd:simpleType>
    </xsd:element>
    <xsd:element name=”exchange”>
    <xsd:simpleType>
                <xsd:restriction base=”xsd:string”>
            <xsd:pattern value=”d\{3)”/>
        </xsd:restriction>
    </xsd:simpleType>
    </xsd:element>
    <xsd:element name=”number”>
    <xsd:simpleType>
                <xsd:restriction base=”xsd:string”>
            <xsdpattern value=”d\{4}”/>
        </xsd:restriction>
    </xsd:simpleType>
    </xsd:element>
</xsd:complexType>
Instance of telephoneNumber:
<telephoneNumber>
<area>123</area>
<exchange>234</exchange>
<number>5678></number>
</telephoneNumber>
```

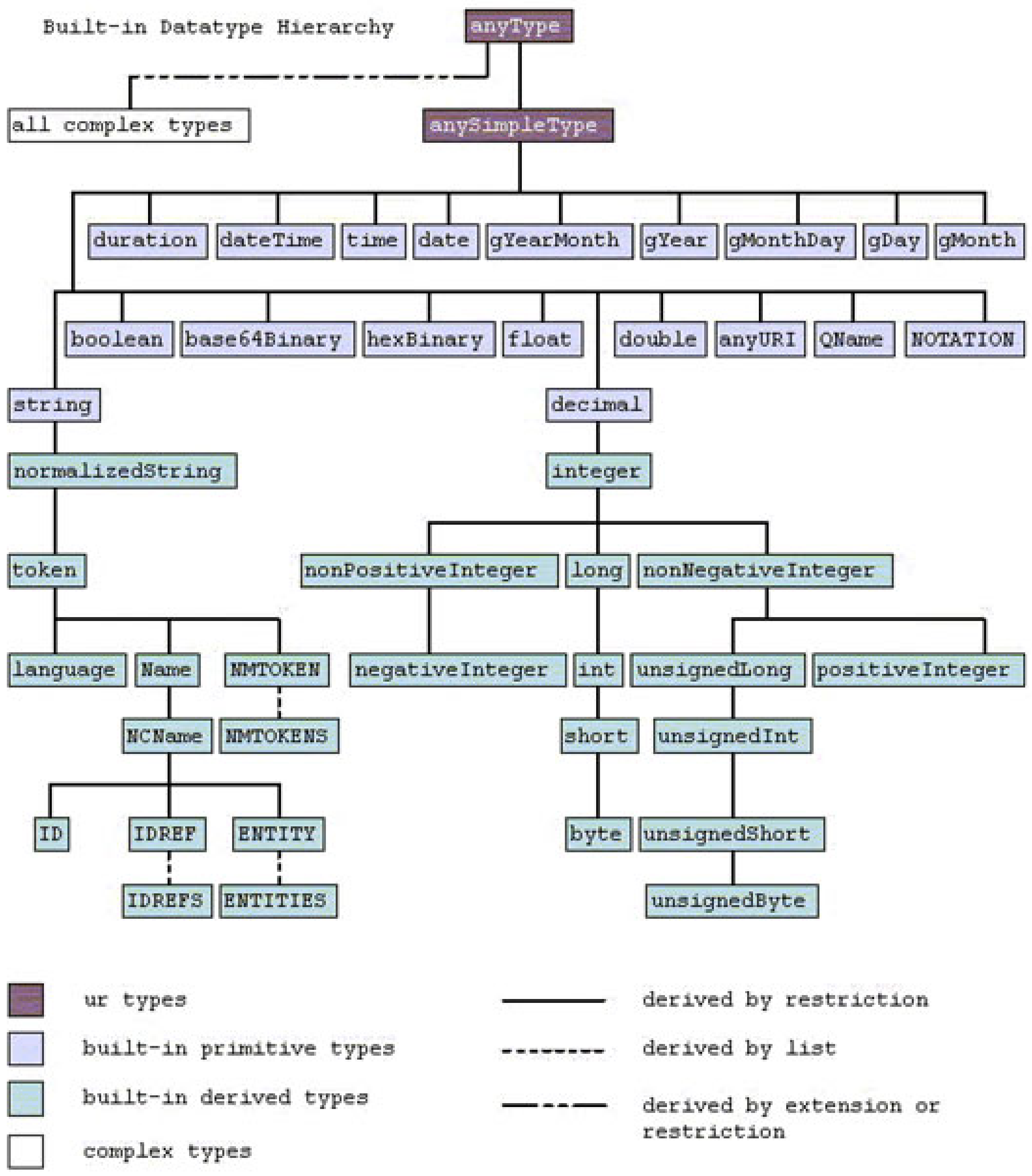

Figure 4.8 XML Schema Data Types (Source: http://www.w3.org/TR/xmlschema-2/ (http://www.w3.org/TR/xmlschema-2/) , Copyright © W3C))

XML Processing/Parsing Models

There are numerous standard processing models available for applications to process/parse XML documents. Three popular ones supported by most computing solution providers are Simple API for XML (SAX) parser, Document Object Model (DOM) Parser, and eXtensible Stylesheet Language Transformation (XSLT) Parser.

SAX Parser provides an event-driven parsing/programming model support; it runs fast with less required resource. DOM Parser provides an in-memory tree-transversal programming model support; it is very flexible and dynamic but consumes much more computing resource (Figure 4.9). Tree walking approach benefits include: 1) Verified as well-formed, 2) Is conformable to a DTD or schema (Validated), and 3) Flexible in terms of data manipulation.

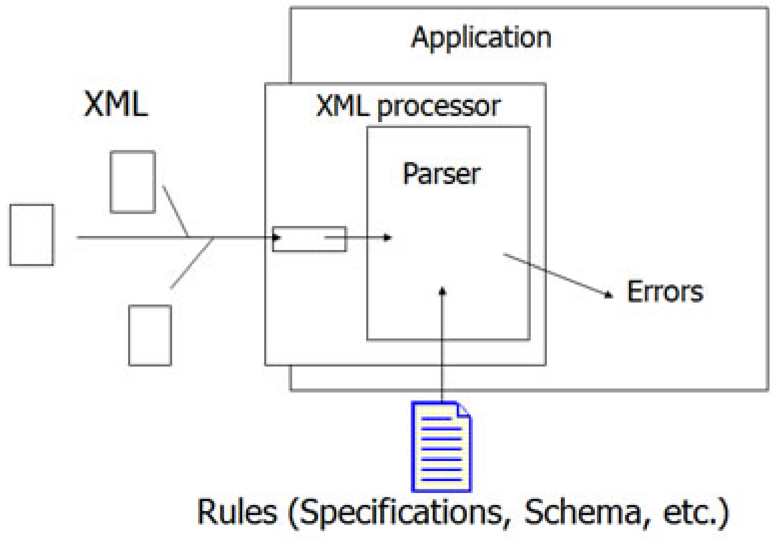

Figure 4.9 Schematic View of XML Document Processing/Parsing

XSLT Parsers enables the support of document transformation from one style to another to meet the reformatting or heterogeneous needs. XSLT standards describe a mechanism for manipulating source XML data into a ***form suitable for publishing or other applications*** (Figure 4.10). XSLT is essentially able to:

- Add prefix or suffix to the content
- Remove, create, reorder and sort elements
- Reuse elements elsewhere in the document
- Transform data from one XML format to another XML format, or to HTML
- Specify the XSL formatting objects to be applied to each element class

XPath acts as a sublanguage within an XSLT stylesheet. An Xpath expression may be used for numerical calculations or string manipulations, or for testing Boolean conditions, but its most characteristic use is to identify parts of the input document to be processed [15].

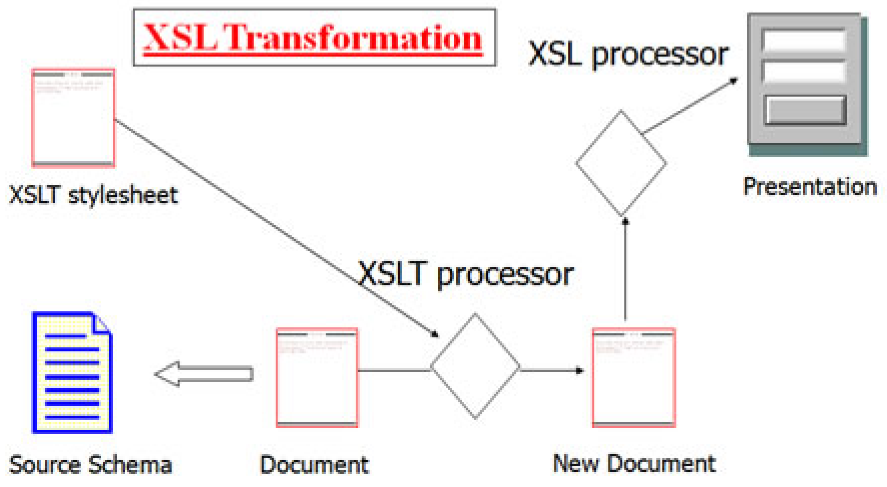

Figure 4.10 Schematic View of XML Document Transformation Processing/Parsing

Utility Package and Samples: Please check the discussion Forum to see more links to parsers and code examples.

SOAP

Simple Object Access Protocol (SOAP) is an XML-based messaging protocol, using a structured and well-defined message and rules for data types to interact between peer applications in a decentralized and distributed computing environment. A SOAP message is essentially an ordinary XML document that typically has the following elements: an Envelope element indicating the XML document as a SOAP message, a Header element containing the message header information (e.g., message routing instruction), a Body element containing request and response information (e.g., data and sharing instruction), and a Fault element illustrating errors and status information. Unless some specific namespaces declared, the default namespace for the SOAP envelope is http://www.w3.org/2001/12/soap-envelope (http://www.w3.org/2001/12/soap-envelope) and the default namespace for SOAP encoding and data types is http://www.w3.org/2001/12/soap-encoding (http://www.w3.org/2001/12/soap-encoding) (Listing 4.4) [5]. Different from RPC, DCOM, and CORBA, SOAP messages can rely on a variety of popular application layer protocols (HTTP, HTTPS, FTP, SMTP, etc.) to allow applications to interact with each other.

Put it simply, here come the SOAP Highlights [4, 5] (Figure 4.11):

- SOAP is an interactive messaging protocol and stands for Simple Object Access Protocol
- SOAP is an XML-based format for exchanging messages between peer applications
- SOAP communicates over a variety of transport and application layer protocols, popularly via Internet
- SOAP is platform- and language-independent, and recommended by W3C

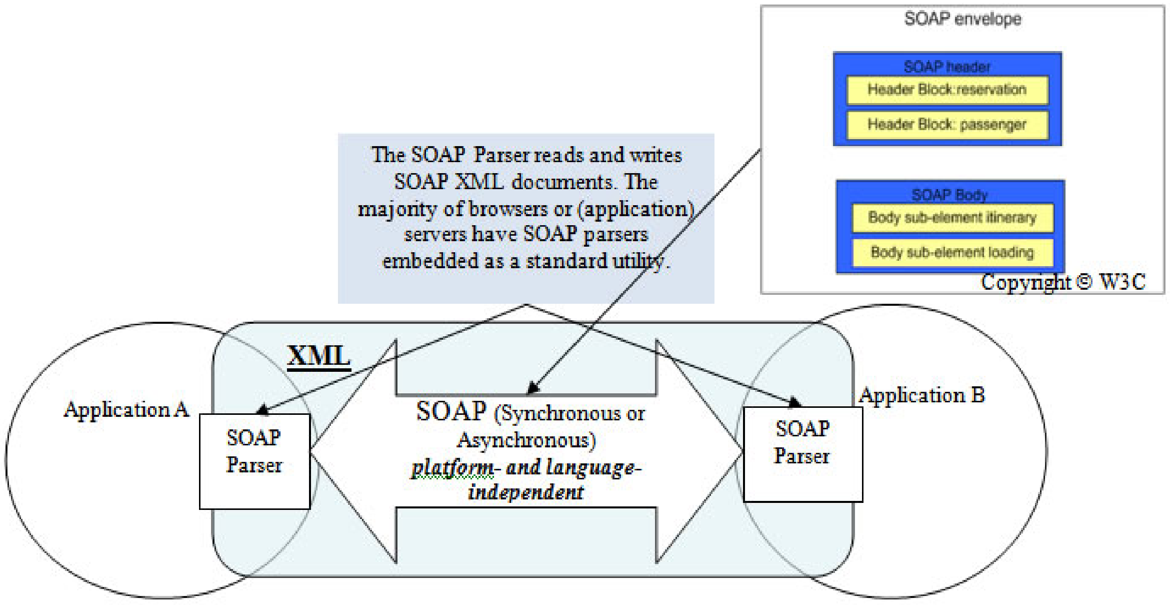
SOAP envelope
SOAP header
Header Block:reservation
Header Block: passenger
SOAP Body
Body sub-element itinerary
Body sub-element loading
Copyright © W3C
The SOAP Parser reads and writes SOAP XML documents. The majority of browsers or (application) servers have SOAP parsers embedded as a standard utility.
XML
Application A
SOAP Parser
SOAP (Synchronous or Asynchronous)
platform- and language-independent
Application B
SOAP Parser

Figure 4.11: A SOAP Message Used to Exchange Structured Information between Application

The SOAP Parser as a standard unitlity built in browsers, servers, and language compilers/interpreters. It reads, writes, and manipulates SOAP XML documents. When accessing the XML document, the SAX-like Parser has all the logic for traversing a SOAP packet, including Envelope, Header, Body, and Fault, dealing with namespaces and tracking down references (Figure 4.11). Listing 4.4 also shows a RPC-style example using SOAP over HTTP.

Listing 4.4 Soap Message Structure

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/l2/soap-envelope”
soap:encodingStyle=”http://www.w3.org/2001/12/soap-encoding”>
<soap:Header>
...
</soap:Header>
<soap:Body>
...
<soap:Fault>
...
</soap:Fault>
</soap:Body>
</soap:Envelope>

Request:
POST/EnterpriseIntegrationWS/StudentRoster.asmx HTTP/1.1
Host:localhost
Content-Type:text/xml; charset=utf-8
Content-Length:length
SOAPAction: “http://Enterpriselntegration.org/NewStudentContactData”

<?xml version=”1.0” encoding=”utf-8"?>
<soap:Envelope xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/200l/XMLSchema”
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <NewStudentContactData xmlns="http://EnterpriseIntegration.org/”>
            <StudID>stríng</StudID>
            <Name>string</Name>
            <SSN>string</SSN>
            <EmailAddress>string</EmailAddress>
            <HomePhone>string</HomePhone>
            <HomeAddr>string</HomeAddr>
            <LocalAddr>string</LocalAddr>
            <EmergencyContact>string</EmergencyContact>
            <ProgramID>string</ProgramID>
            <PaymentID>string</PaymentID>
            <AcademicStatus>string</AcademicStatus>
        </NewStudentContactData>
    </soap:Body>
</soap:Envelope>


Response:

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

<?xml version=”1.0” encoding=”utf-8"?>
<soap:Envelope xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/”>
    <soap:Body>
        <NewStudentContactDataResponse xmlns=”http://EnterpriseIntegration.org/">
            <NewStudentContactDataResult>string</NewStudentContactDataResult>
        </NewStudentContactDataResponse>
    </soap:Body>
</soap:Envelope>
```

Utility Package and Samples: Please check the Software Help Forum to see more links to parsers and code examples.

WSDL

WSDL is an XML format specification for describing provided computational services as a set of endpoints operating on messages that can be constructed as either document-oriented or procedure-oriented information. Similar to the IDL discussed in Lesson 3, the operations and messages as the service endpoint are described abstractly (i.e., the Abstract or Service Definition Section in Figure 4.12), and then bound to a concrete network protocol and message format to define its endpoint (i.e., the Concrete or Binding Section in Figure 4.12). WSDL is extensible to allow description of endpoints and their messages regardless of network protocols are used to communicate. The popular bindings are WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME [17].

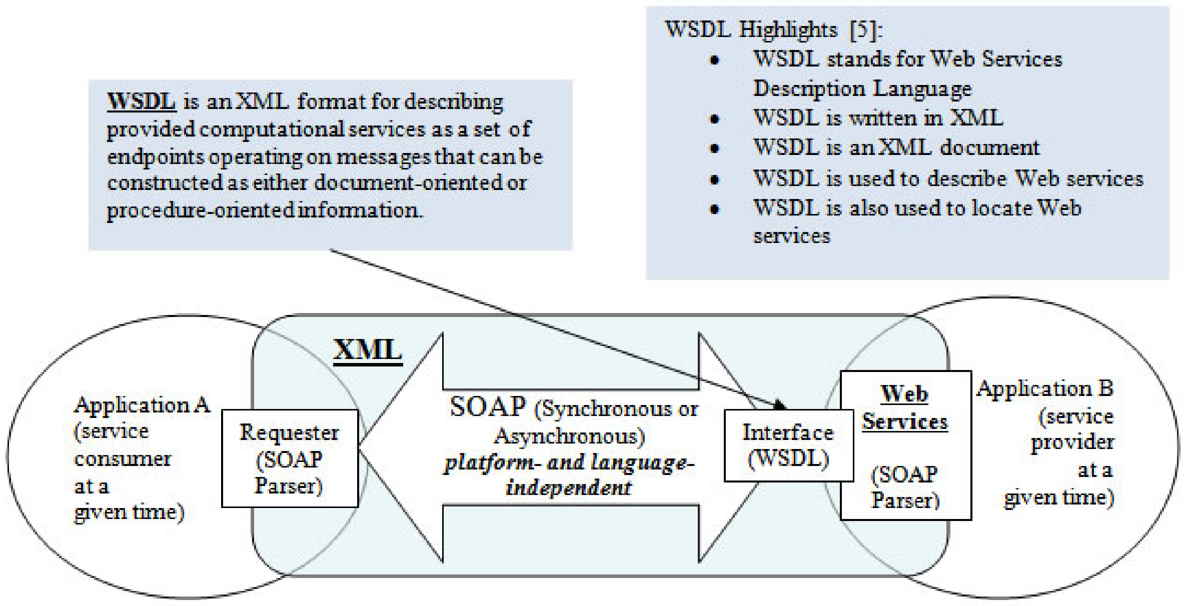
WSDL Highlights [5]:
• WSDL stands for Web Services Description Language
• WSDL is written in XML
• WSDL is an XML document
• WSDL is used to describe Web services
• WSDL is also used to locate Web services
WSDL is an XML format for describing provided computational services as a set of endpoints operating on messages that can be constructed as either document-oriented or procedure-oriented information.
XML
Application A (service consumer at a given time)
Requester (SOAP Parser)
SOAP (Synchronous or Asynchronous) platform- and language-independent
Interface (WSDL)
Web Services
(SOAP Parser)
Application B (service provider at a given time)

Figure 4.11: A WSDL: Formal Description of the Service Interface

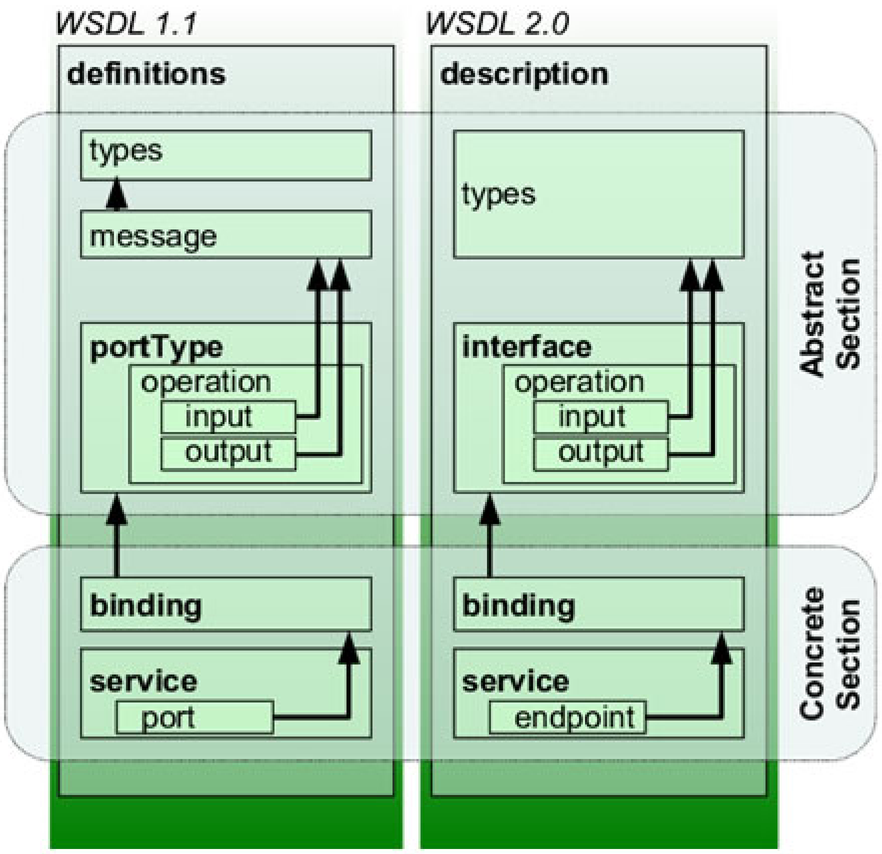

Figure 4.12: WSDL (Objects) Model (Source: Wikipedia [2], Copyright © http://en.wikipedia.org/ (http://en.wikipedia.org/wiki/Main_Page)

WSDL (http://www.w3.org/TR/wsdl) is an XML-based language that allows formal descriptions of the **interfaces of Web Services**. It essentially answers the following questions:

- which **interaction** does the service provide?
- which **arguments and results** are involved in the interactions?
- which network addresses are used to **locate** the service?
- which communication **protocol** should be used?
- which **data formats** are the messages represented in?

So what are the benefits?

- an interface description is a **contract** between the server developers and the client developers
- having **formal** descriptions allows **tool support**, e.g. code template generators (see Microsoft Studio, the Apache AXIS project, the alphaWorks Emerging Technologies Toolkit, and CapeClear's WSDL tools)

An actual WSDL document consists of a set of definitions of the following kinds (Figure 4.12):

- types - containing XML Schema element and type definitions
- message - consisting of either
  - a number of named parts typed by XML Schema elements, or

  - a single part typed by a XML Schema type

- portType (Interface) - describing a set of operations, each being either
  - *one-way*: receiving an input message,
  - *request-response*: receiving an input message and then responding with an output message (like Remote Procedure Calls),
  - *solicit-response*: sending an output message and then receiving an input message, or
  - *notification*: sending an output message
- binding - selects communication protocol and data formats for each operation and message
- service - describes a collection of named ports, each associated with a binding and a network address

In addition, an import mechanism allows modularization of WSDL definitions.

Utility Package and Samples: Please check the Software Help Forum to see more links to editors and code examples. WSDLs now can be easily generated from the tools you like to use, for example, Microsoft Studio, the Apache AXIS project, the alphaWorks Emerging Technologies Toolkit, and CapeClear's WSDL tools [18]. You have to get familiar with the integrated development environment (IDE). Then you can simply create a project by creating some simple classes. When you compile these classes, you will have their WSDLs right away. I used Microsoft Studio 2005 to create a Web Service listed in Listing 4.5

Listing 4.5 Sample of WSDL for Student Contact Information (generated by MS Studio 2005)

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://EnterpriseIntegration.org/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://EnterpriseIntegration.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
- <wsdl:types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://EnterpriseIntegration.org/">
- <s:element name="NewStudentContactData">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="StudID" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="SSN" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EmailAddress" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="HomePhone" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="HomeAddr" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="LocalAddr" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EmergencyContact" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="ProgramID" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="PaymentID" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="AcademicStatus" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="NewStudentContactDataResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="NewStudentContactDataResult" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
  </s:schema>
  </wsdl:types>
- <wsdl:message name="NewStudentContactDataSoapIn">
  <wsdl:part name="parameters" element="tns:NewStudentContactData" />
  </wsdl:message>
- <wsdl:message name="NewStudentContactDataSoapOut">
  <wsdl:part name="parameters" element="tns:NewStudentContactDataResponse" />
  </wsdl:message>
- <wsdl:portType name="StudentRosterSoap">
- <wsdl:operation name="NewStudentContactData">
  <wsdl:input message="tns:NewStudentContactDataSoapIn" />
  <wsdl:output message="tns:NewStudentContactDataSoapOut" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="StudentRosterSoap" type="tns:StudentRosterSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
```

```
- <wsdl:operation name="NewStudentContactData">
  <soap:operation soapAction="http://EnterpriseIntegration.org/NewStudentContactData" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="StudentRosterSoap12" type="tns:StudentRosterSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="NewStudentContactData">
  <soap12:operation soapAction="http://EnterpriseIntegration.org/NewStudentContactData" />
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="StudentRoster">
- <wsdl:port name="StudentRosterSoap" binding="tns:StudentRosterSoap">
  <soap:address location="http://localhost/EnterpriseIntegrationWS/StudentRoster.asmx" />
  </wsdl:port>
- <wsdl:port name="StudentRosterSoap12" binding="tns:StudentRosterSoap12">
  <soap12:address location="http://localhost/EnterpriseIntegrationWS/StudentRoster.asmx" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

After you learn XML, SOAP, and WSDL, the best way to master the Web Services technology is to develop some examples by yourself.

UDDI (10 of 11)

UDDI

The UDDI specifications and schema are used to build discovery services on the Internet. These discovery services provide a consistent publishing interface and allow programmatic discovery of services (Figure 4.13).

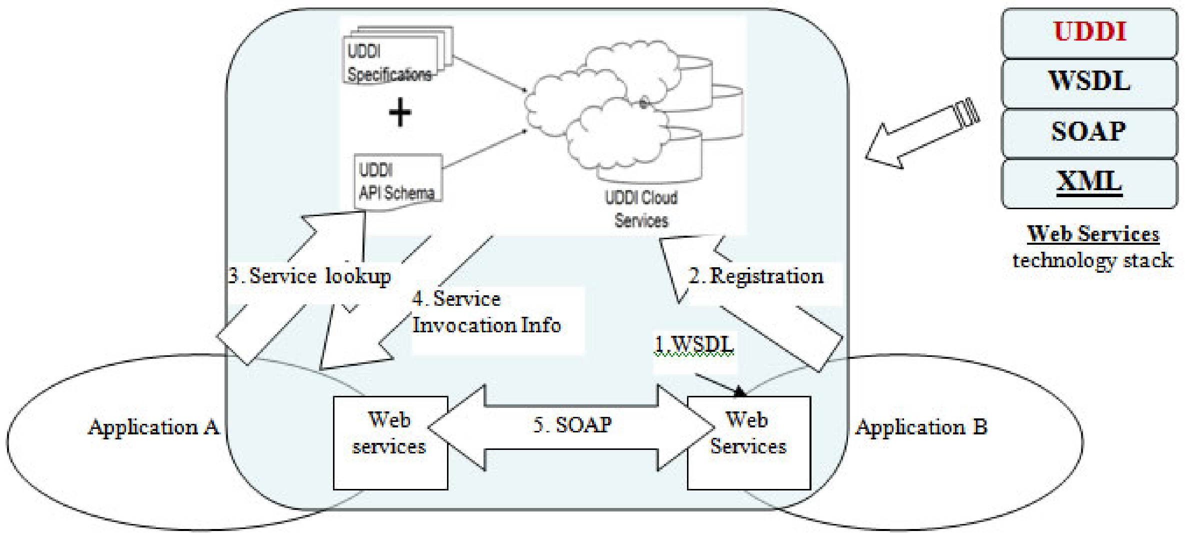

Figure 4.13: UDDI as the Depository of Service Registration and Lookup

UDDI is essentially a comprehensive yellow page for web services. Different business partners can register public information about their Web services and types of services with UDDI, and applications can view information about these Web services with UDDI (Figure 4.13). In general UDDI provides the following information:

- *Company Contact Information*: any kind of basic business contact information
- *Categorized Businesses by Standard Taxonomies*: the kind of information a service provides
- *Technical Information about Services that are Exposed*: business rules and instructions specifying how to invoke Web services.

References (11 of 11)

References

[1] Roshen, W. 2009. SOA-based Enterprise Integration: A Step-by-Step Guide to Services-based Application Integration. McGraw-Hill, New York, USA.

[2] Wikipedia:

- JRMP: http://en.wikipedia.org/wiki/Java_Remote_Method_Protocol,
- XML: http://en.wikipedia.org/wiki/XML,
- SOAP: http://en.wikipedia.org/wiki/SOAP,
- WSDL: http://en.wikipedia.org/wiki/Web_Services_Description_Language,
- UDDI: http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration.

[3] Qiu, R. Information Technology as a Service. Enterprise Service Computing: From Concept to Deployment. (Chapter 1) Ed. By R. Qiu. IGI Publishing, Hershey, PA.

[4] W3C: http://www.w3.org/.

[5] W3School. http://www.w3schools.com/.

[6] Description of W3C Technology Stack Illustration: http://www.w3.org/Consortium/techstack-desc.html.

[7] Extensible Markup Language (XML): http://www.w3.org/XML/.

[8] XML Specification 1.0 (Fifth Version): http://www.w3.org/TR/REC-xml/.

[9] XML Spy: http://www.altova.com/xml-editor/.

[10] <oXygen/> XML Editor: http://www.oxygenxml.com/.

[11] Open Source XML Editor: http://www.syntext.com/products/serna-free/.

[12] XML Schema: http://www.w3.org/XML/Schema, http://www.w3.org/standards/xml/schema.

[13] XML Schema Tutorial: http://www.liquid-technologies.com/Tutorials/XmlSchemas/XsdTutorial_01.aspx.

[14] Microsoft MSDN XML: http://msdn.microsoft.com/en-us/data/bb190600.aspx,

- Sun Java: http://java.sun.com/webservices/index.jsp,
- Apache Xerces2 Java Parser: http://xerces.apache.org/xerces2-j/.

[15] XPath Language: http://www.w3.org/TR/xpath/.

[16] SOAP: http://www.w3.org/TR/soap/.

[17] Web Services Description Language (WSDL) 1.1: http://www.w3.org/TR/wsdl, http://www.w3.org/TR/wsdl20/.

[18] Web Services Utility

- Implement Web Services with the Windows Web Services API: http://code.msdn.microsoft.com/wwsapi,
- Apache Axis: xml.apache.org/axis,
- ETTK (Emerging Technologies Toolkit): www.alphaworks.ibm.com/tech/ettk,
- Java EE Web Services Technologies: http://java.sun.com/javaee/technologies/webservices/,
- Java SE Technologies and Web Services: http://java.sun.com/javase/technologies/webservices.jsp

[19] UDDI Version 3.0.2: http://www.uddi.org/pubs/uddi_v3.htm

Other Cool Web Service Websites:

- XML.com's Web service section: webservices.xml.com

- W3C's Web Services activity: www.w3.org/2002/ws
- Apache's Web Service project: ws.apache.org
- Web Service links and resources: wsindex.org
- TopXML XML Code Library: http://www.xml.com/pub/r/1351