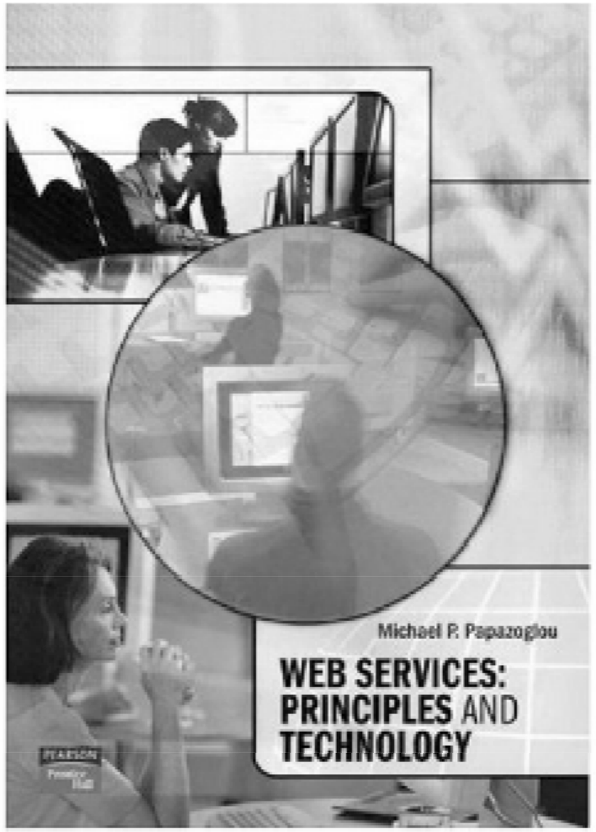
Web Services: Principles & Technology

Chapter 3 Brief Overview of XML

Mike P. Papazoglou &
mikep@uvt.nl



UNIVERSITEIT VAN TILBURG

Topics

- ***XML document structure***
- *XML schemas reuse*
- *Document navigation and transformation*

Markup

- A method of distinguishing text from instructions in typesetting systems.
- Markup = instructions to computerized typesetting systems.
- Special characters are used to show where a markup instruction starts and stops.

Example:

```
<centre on> This is a <italics on> very serious <italics
off> matter.<centre off>
```

This is a *very serious* matter.

“Tags” are easily understandable by both the human reader and the machine.

XML

- **Extensible Markup Language** (XML) is not a set of tags itself but a ***meta-language*** that allows you to create and use tag sets in a standard way.

- Provides fixed rules about markup structures (elements, attributes, and entities), about their notation, and their function.

Book Catalog in XML

```
<BOOK>
    <TITLE>
      Web Services: Principles and
    Technology
   </TITLE>
    <AUTHOR>
      Mike P. Papazoglou
    </AUTHOR>
    <DATE> 2007 </DATE>
    <PUBLISHER>
      Prentice Hall
   </PUBLISHER>
</BOOK>
```

This document has XML tags.
A human and a computer can now easily extract the publisher data.

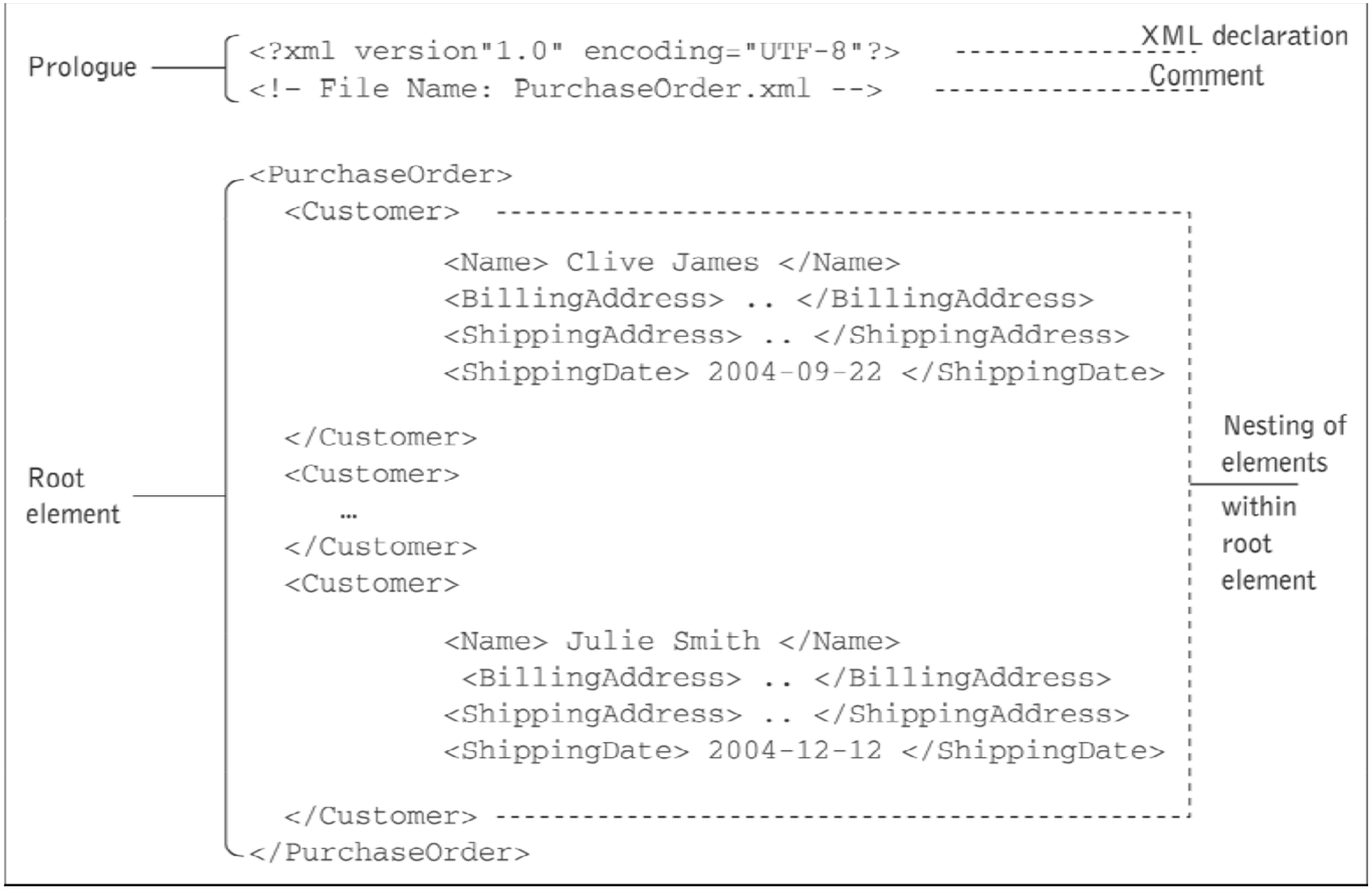
XML structure

- **Document type:** XML documents are regarded as having types.
  - XML's constituent parts and their structure formally define the type of a document.
- An XML document is composed of named **containers** and their contained data values.
  - containers are represented as declarations, elements, and attributes.
- An XML document is also known as an **instance or XML document instance** to signify that it represents one possible set of data for a particular markup language.

Example of an XML document instance.

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

<BillingInformation>
    <Name> Right Plastic Products  </Name>
    <BillingDate> 2002-09-15 </BillingDate>
    <Address>
      <Street> 158 Edward st. </Street>
      <City> Brisbane </City>
      <State> QLD </State>
      <PostalCode> 4000 </PostalCode>
    </Address>
</BillingInformation>
```

Layout of a typical XML document



```
<?xml version"1.0" encoding="UTF-8"?>
<!- File Name: PurchaseOrder.xml -->

<PurchaseOrder>
  <Customer>
          <Name> Clive James </Name>
          <BillingAddress> .. </BillingAddress>
          <ShippingAddress> .. </ShippingAddress>
          <ShippingDate> 2004-09-22 </ShippingDate>
  </Customer>
  <Customer>
     …
  </Customer>
  <Customer>
          <Name> Julie Smith </Name>
           <BillingAddress> .. </BillingAddress>
          <ShippingAddress> .. </ShippingAddress>
          <ShippingDate> 2004-12-12 </ShippingDate>
  </Customer>
</PurchaseOrder>
```

Layout of a typical XML document

XML: Elements

- Elements are fundamental units of content comprising element name and element content.
  - An **element** is a sequence of characters that begins with a start tag and ends with an end tag and includes everything in between.

```
<chapter number="1">
        Text for Chapter 1
</chapter>
```

- **What is content?**
  The characters in between the tags (rendered in green in this presentation) constitute the **content**.

- The topmost element of the XML document is a single element known as the **root element**.
- Elements contained in other elements are referred to as **nested elements**.
  - The containing element is the **parent element** and the nested element is called the **child element**.

XML: Attributes

- Another way of putting data into an XML document is by adding *attributes* to start tags.
  - An attribute specification is a name–value pair that is associated with an element.
  - Attributes are used to better specify the content of an element on which they appear by adding information about a defined element.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<BillingInformation customer-type=”manufacturer”>
    <Name> Right Plastic Products </Name>
    <BillingDate> 2002-09-15 </BillingDate>
    <Address>
      <Street> 158 Edward st. </Street>
      <City> Brisbane </City>
      <State> QLD </State>
      <PostalCode> 4000 </PostalCode>
    </Address>
</BillingInformation>
```

XML Namespaces

- Namespaces in XML provide a facility for associating the elements and/or attributes in all or part of a document with a particular schema and avoiding name clashes.
- Namespace declarations have a scope.
  - A namespace declaration is in scope for the element on which it is declared and of that element's children.
- The namespace name and the local name of the element together form a globally unique name known as a *qualified name*.

```
<?xml version="1.0" encoding="UTF-8"?>
<BillingInformation customer-type="manufacturer"
     xmlns="http://www.plastics_supply.com/BillInfo">
    <Name> Right Plastic Products </Name>
    <Address xmlns="http://www.plastics_supply.com/Addr">
      <Street> 158 Edward st. </Street>
      <City> Brisbane </City>
      <State> QLD </State>
      <PostalCode> 4000 </PostalCode>
    </Address>
    <BillingDate> 2002-09-15 </BillingDate>
</BillingInformation>
```

Uniform resource identifier (URI).

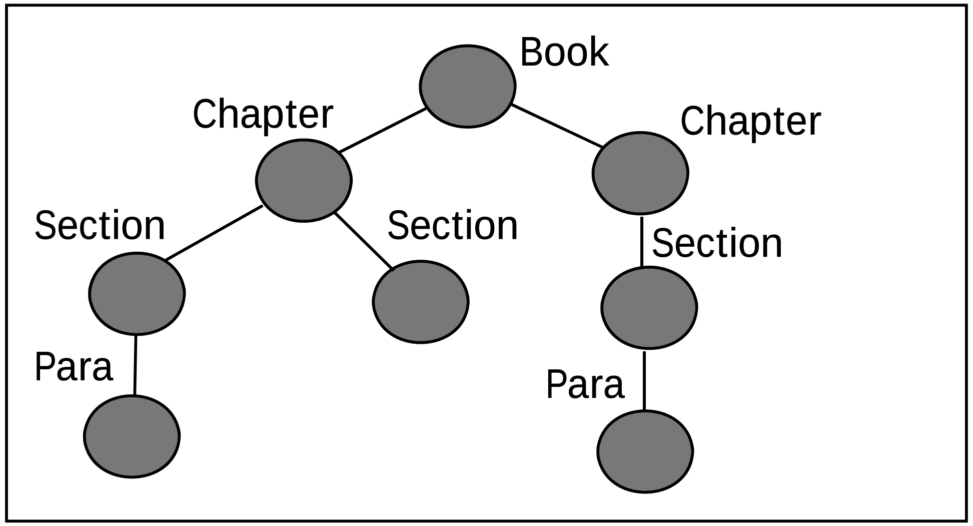
XML: structure

- An XML document **must have a root** tag.
- An XML document is an information unit that can be seen in two ways:
  - As a **linear sequence of characters** that contain characters data and markup.
  - As an abstract data structure that is a **tree of nodes.**

Linear sequence

```
<book>
  <chapter n="1">   Title 1 </chapter>
  <section n="1.1"> Section 1.1 </section>
  <paragraph> .... <paragraph>
  <section n="1.2"> Section 1.2 </section>
  <chapter n="2">   Title 2 </chapter>
  <section n="2.1"> Section 2.1 </section>
  < paragraph > .... < paragraph >
</book>
```

Tree



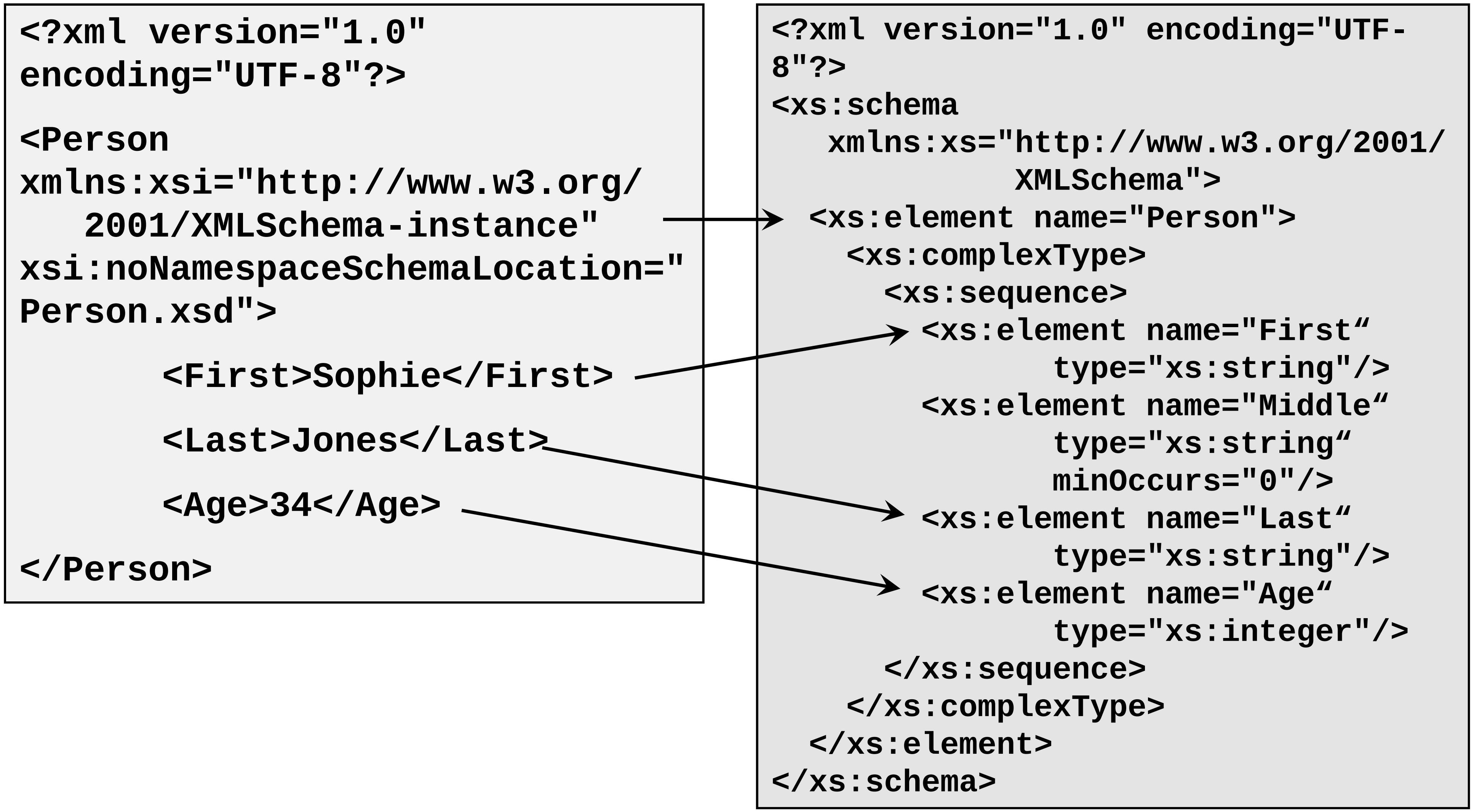
XML Schema

- **XML schema** refers to a document that defines the content of and structure for expressing XML documents.
- Schemas provide support for additional meta-data characteristics such as structural relationships, cardinality, valid values, and data types.
- Each type of schema acts as a method of describing data characteristics and applying rules and constraints to a referencing XML document.
- An XML schema describes the elements and attributes that may be contained in a schema conforming document and the ways that the elements may be arranged within a document structure.

XML schema and instance document

File “Person.xml”

File “Person.xsd”

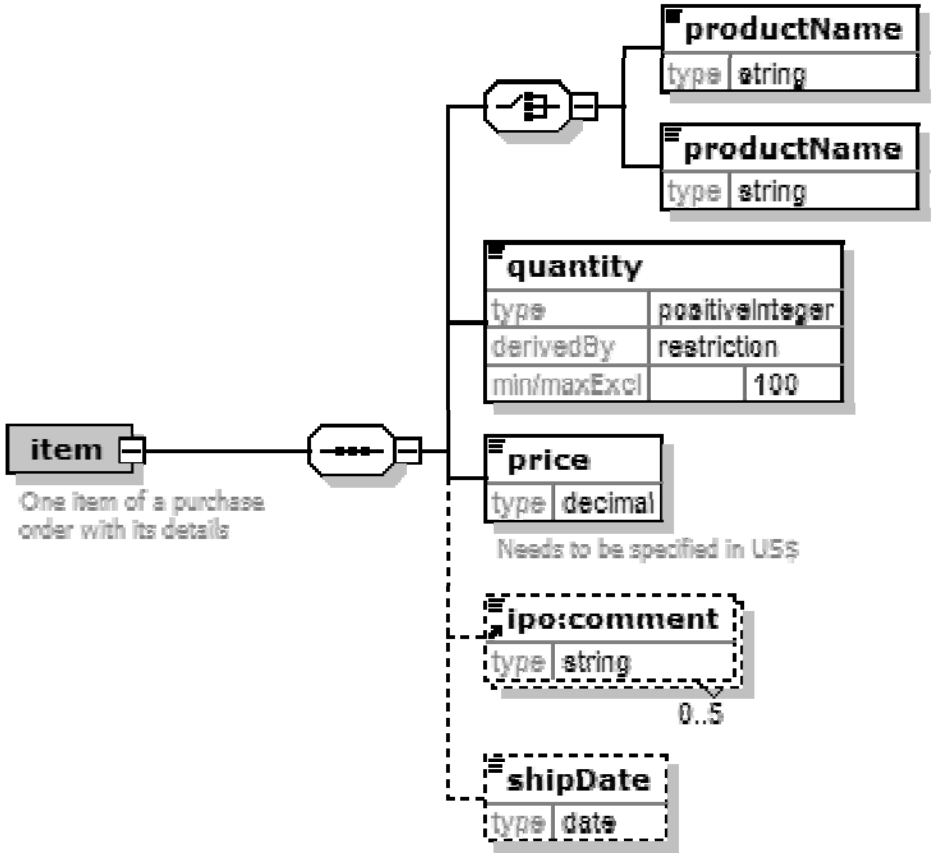


```
<?xml version="1.0"
encoding="UTF-8"?>

<Person
xmlns:xsi="http://www.w3.org/
   2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="
Person.xsd">

      <First>Sophie</First>

      <Last>Jones</Last>

      <Age>34</Age>

</Person>
```

```
<?xml version="1.0" encoding="UTF-
8"?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/
             XMLSchema">
  <xs:element name="Person">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="First“
               type="xs:string"/>
        <xs:element name="Middle“
               type="xs:string“
               minOccurs="0"/>
        <xs:element name="Last“
               type="xs:string"/>
        <xs:element name="Age“
               type="xs:integer"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schema Components

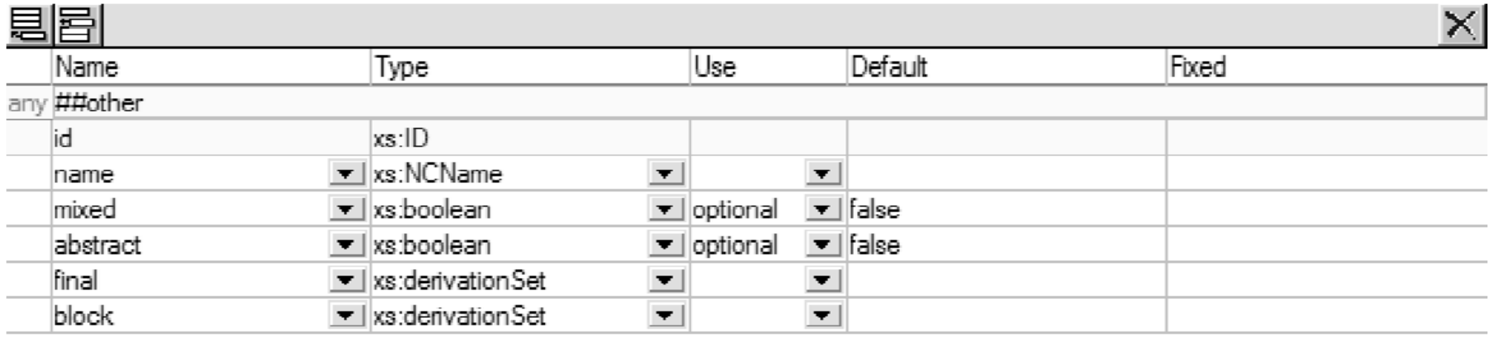
- **Elements and their content model**



```
<element name="item">
  <annotation>
    <documentation>One item of a purchase order with its
                  details</documentation>
  </annotation>
  <complexType>
    <sequence>
      <choice>
        <element name="productName" type="string"/>
        <element name="productName" type="string"/>
      </choice>
      <element name="quantity">
        <simpleType>
          <restriction base="positiveInteger">
            <maxExclusive value="100"/>
          </restriction>
        </simpleType>
      </element>
      <element name="price" type="decimal">
        <annotation>
          <documentation>Needs to be specified in
                        US$</documentation>
        </annotation>
      </element>
      <element ref="ipo:comment" minOccurs="0" maxOccurs="5"/>
      <element name="shipDate" type="date" minOccurs="0"/>
    </sequence>
    <attribute name="partNum" type="ipo:Sku"/>
  </complexType>
</element>
```

Schema Components (continued)

- **Attributes and Attribute Groups**



	Name	Type	Use	Default	Fixed
any	##other				
	id	xs:ID			
	name	xs:NCName			
	mixed	xs:boolean	optional	false	
	abstract	xs:boolean	optional	false	
	final	xs:derivationSet			
	block	xs:derivationSet			

```
<xs:attribute name="name" type="xs:NCName"/>
<xs:attribute name="mixed" type="xs:boolean" use="optional" default="false"/>
<xs:attribute name="abstract" type="xs:boolean" use="optional" default="false"/>
<xs:attribute name="final" type="xs:derivationSet"/>
<xs:attribute name="block" type="xs:derivationSet"/>
```

Schema Components (continued)

- **Complex Types**

```
<xsd:complexType name="PersonType">
 <xsd:sequence>
  <xsd:element name="First" type="xsd:string"/>
  <xsd:element name="Last" type="xsd:string"/>
  <xsd:element name="Title" type="xsd:string“
               minOccurs="0"/>
  <xsd:element name="PhoneExt" type="xsd:int"/>
  <xsd:element ref="EMail"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:element name="Person" type="PersonType"/>
<xsd:element name="VIP“
             substitutionGroup="Person">
 <xsd:complexType>
  <xsd:complexContent>
   <xsd:extension base="PersonType">
    <xsd:attribute name="IQ" type="xsd:int"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:element>
```

Topics

- *XML document structure*
- ***XML schemas reuse***
- *Document navigation and transformation*

Complex type extensions

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

<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:PO="http://www.plastics_supply.com/PurchaseOrder"
  targetNamespace="http://www.plastics_supply.com/PurchaseOrder">

          <xsd:complexType name="Address">
             <xsd:sequence>
                 <xsd:element name="Number" type="xsd:decimal"/>
                 <xsd:element name="Street" type="xsd:string"/>
                 <xsd:element name="City" type="xsd:string" minOccurs="0"/>
             </xsd:sequence>
          </xsd:complexType>

          <xsd:complexType name="AustralianAddress">
              <xsd:complexContent>
                  <xsd:extension base="PO:Address">
                    <xsd:sequence>
                      <xsd:element name="State" type="xsd:string"/>
                      <xsd:element name="PostalCode" type="xsd:decimal"/>
                      <xsd:element name="Country" type="xsd:string"/>
                    </xsd:sequence>
                  </xsd:extension>
              </xsd:complexContent>
          </xsd:complexType>
</xsd:schema>
```

Complex type restrictions

```
<!-- Uses the data type declarations from Listing on page 17 -->
 <xsd:complexType name="AustralianPostalAddress">
    <xsd:complexContent>
       <xsd:restriction base="PO:AustralianAddress">
              <xsd:sequence>
                 <xsd:element name="Number" type="xsd:decimal"/>
                 <xsd:element name="Street" type="xsd:string"/>
                 <xsd:element name="City" type="xsd:string" minOccurs="0" maxOccurs="0"/>
                 <xsd:element name="State" type="xsd:string"/>
                 <xsd:element name="PostalCode" type="xsd:decimal"/>
              <xsd:element name="Country" type="xsd:string"/>
           </xsd:sequence>
       </xsd:restriction>
    </xsd:complexContent>
 </xsd:complexType>
```

Complex type polymorphism

```
<!-- Uses the data type declarations from Listing on page 17 -->
  <xsd:complexType name="PurchaseOrder">
    <xsd:sequence>
      <xsd:element name="Name" minOccurs="1" maxOccurs="1">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string"/>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="shippingAddress" type="PO:Address" minOccurs= "1"
                                                            maxOccurs="1"/>
      <xsd:element name="billingAddress" type="PO:Address" minOccurs= "1"
                                                            maxOccurs="1"/>
        <xsd:choice minOccurs="1" maxOccurs="1">
            <xsd:element name="BillingDate"  type="xsd:date"/>
            <xsd:element name="ShippingDate" type="xsd:date"/>
        </xsd:choice>
    </xsd:sequence>
  </xsd:complexType
```

Variant of the PurchaseOrder type that uses the base type Address for its billingAddress and shippingAddress elements.

Complex type polymorphism (Continued)

```
<?xml version="1.0" encoding=”UTF-8”?>
<PO:PurchaseOrder
xmlns:PO="http://www.plastics_supply.com/PurchaseOrder">

    <Name> Plastic Products </Name>
    <shippingAddress xsi:type="PO:AustralianAddress">
      <Number> 459 </Number>
      <Street> Wickham st. </Street>
      <City> Fortitude Valley </City>
      <State> QLD </State>
      <PostalCode> 4006 </PostalCode>
      <Country> Australia </country>
    </shippingAddress>

    <billingAddress xsi:type=="PO:AustralianPostalAddress">
      <Number> 158 </Number>
      <Street> Edward st. </Street>
      <State> QLD </State>
      <PostalCode> 4000 </PostalCode>
      <Country> Australia </Country>
    </billingAddress>
      <BillingDate> 2002-09-15 </BillingDate>
</PO:PurchaseOrder>
```

An instance document can now use any type derived from base type Address for its billingAddress and shippingAddress elements.

PurchaseOrder type uses the derived AustralianAddress type as its billingAddress and the derived AustralianPostalAddress type as its shippingAddress elements.

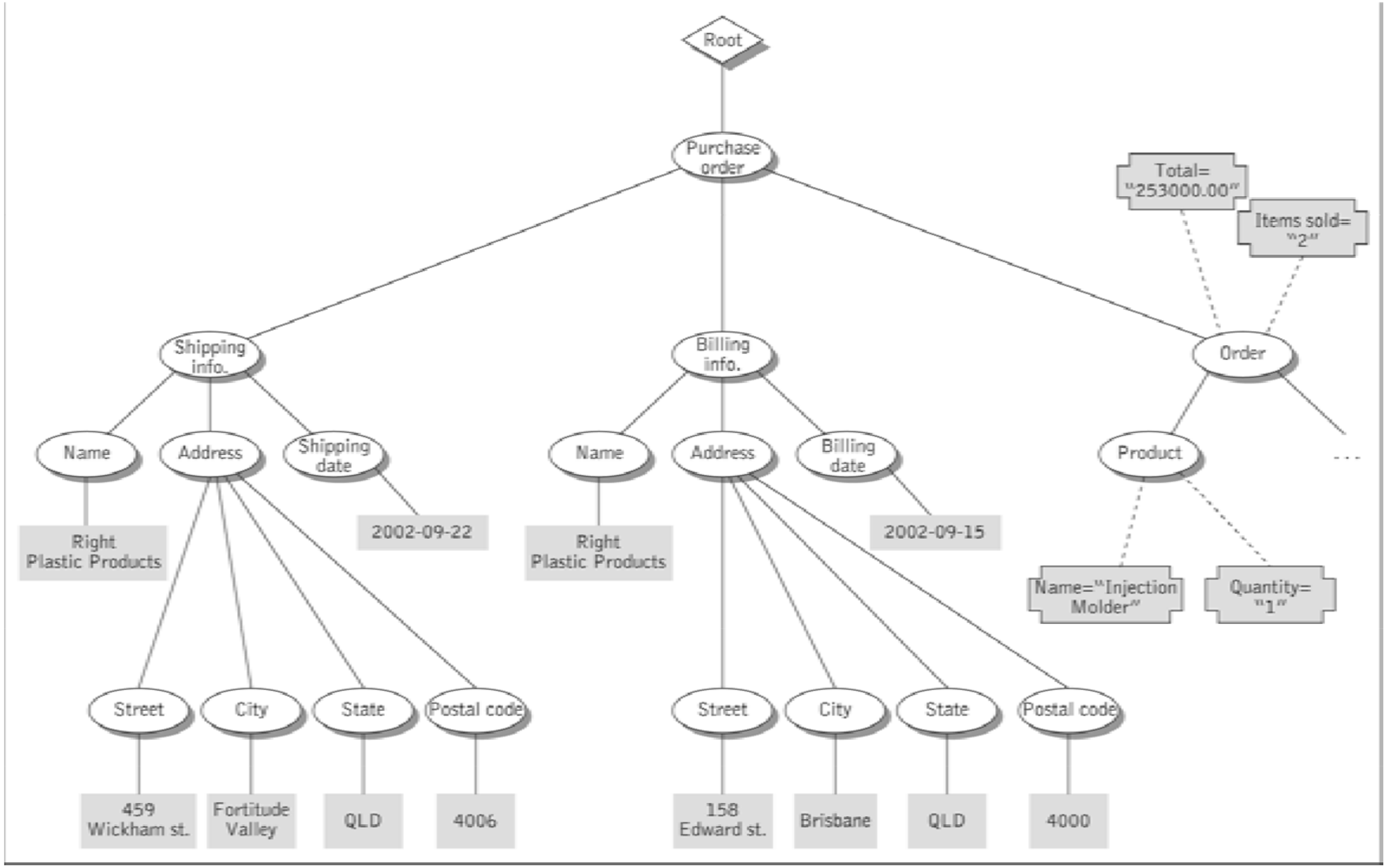
Topics

- *XML document structure*
- *XML schemas reuse*
- ***Document navigation and transformation***

Document Navigation and Transformation

- XML is primarily used to describe and contain data. Although the most obvious and effective use of XML is to describe data, the eXtensible Stylesheet Language Transform (**XSLT**) can also be used to format or transform XML content for presentation to users.
- The XSLT process transforms an XML structure into presentation technology such as HTML or into any other required forms and structures.
- XSLT uses the XML Path Language or **XPath** to address and locate sections of XML documents.
- XPath is a standard for creating expressions that can be used to find specific pieces of information within an XML document.
  - XPath is an abstract language that defines a tree model that codifies the logical structure of an XML document against which all expressions are evaluated.

XPath tree model for instance document



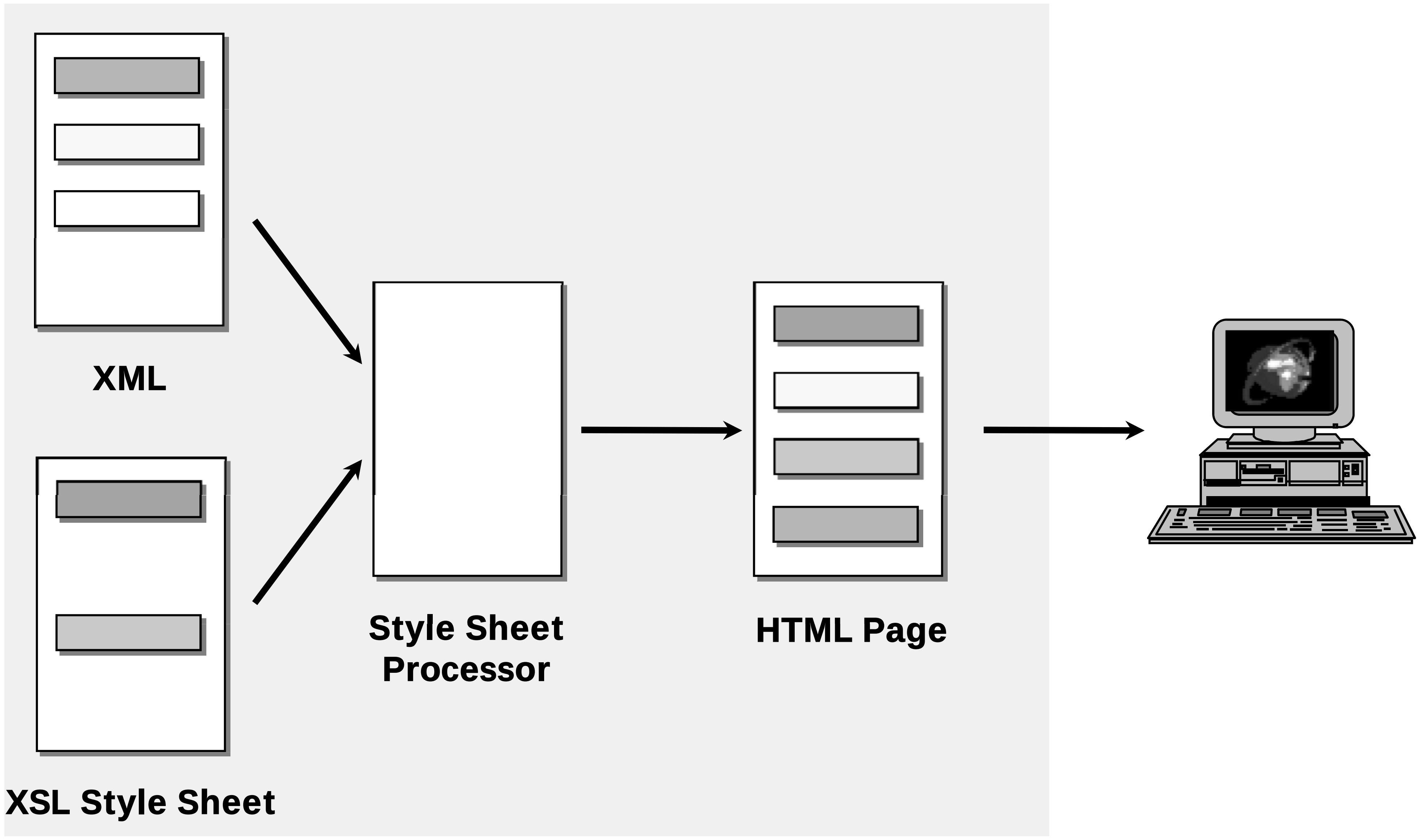
XPath tree model for instance document

An XML instance document conforming to the schema in Listing-3.6

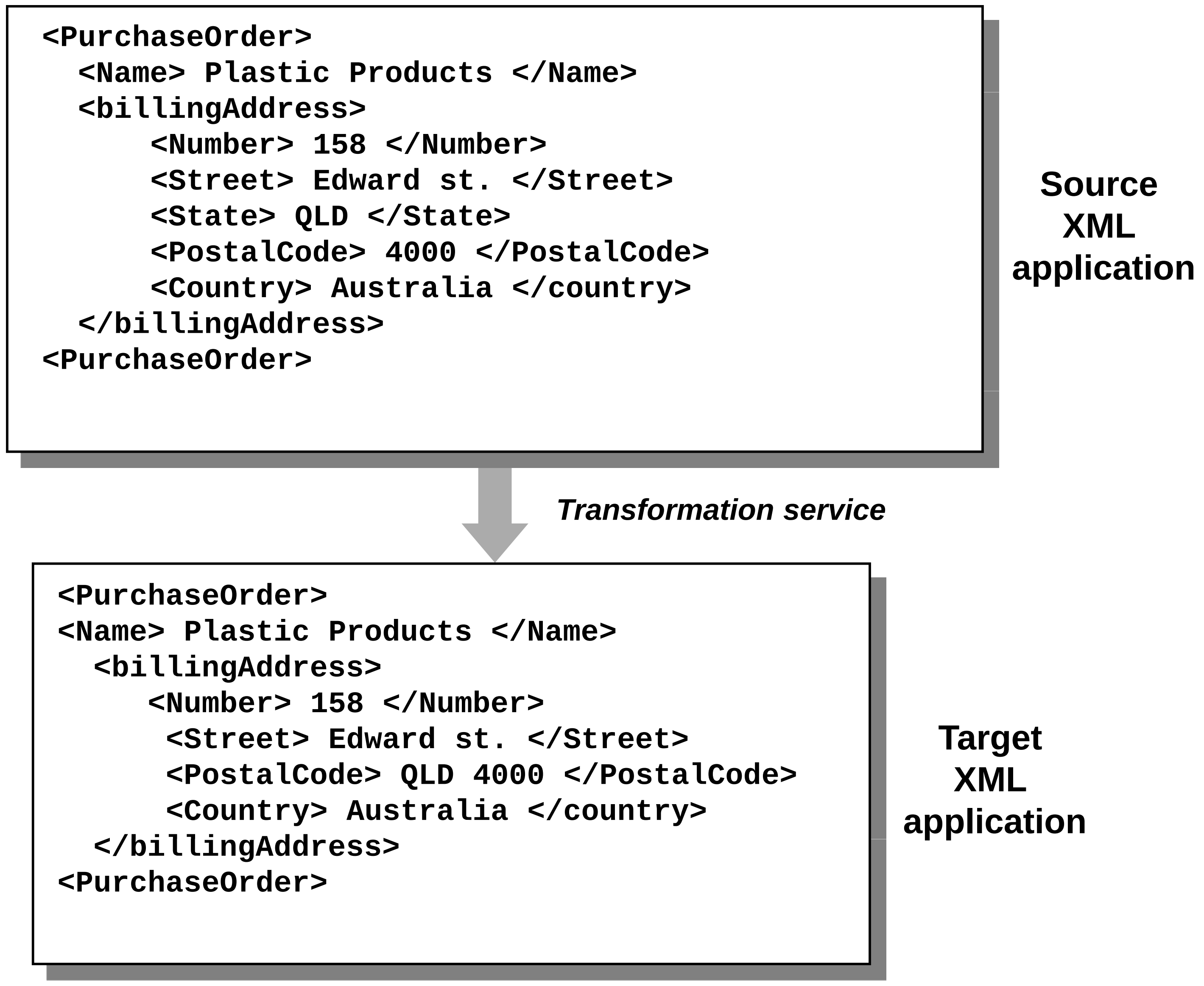
```
<?xml version="1.0" encoding="UTF-8"?>
 <PO:PurchaseOrder
 xmlns:PO="http://www.plastics_supply.com/PurchaseOrder"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.plastics_supply.com/PurchaseOrder purchaseOrder.xsd">

  <ShippingInformation>
    <Name> Right Plastic Products Co. </Name>
    <Address>
      <Street> 459 Wickham st. </Street>
      <City> Fortitude Valley </City>
      <State> QLD </State>
      <PostalCode> 4006 </PostalCode>
    </Address>
     <ShippingDate> 2002-09-22 </ShippingDate>
  </ShippingInformation>

  <BillingInformation>
    <Name> Right Plastic Products Inc. </Name>
    <Address>
      <Street> 158 Edward st. </Street>
      <City> Brisbane </City>
      <State> QLD </State>
      <PostalCode> 4000 </PostalCode>
    </Address>
    <BillingDate> 2002-09-15 </BillingDate>
  </BillingInformation>

  <Order Total="253000.00" ItemsSold="2">
    <Product Name="Injection Molder" Price="250000.00"
     Quantity="1"/>
    <Product Name="Adjustable Worktable" Price="3000.00"
     Quantity="1"/>
  </Order>
</PO:PurchaseOrder>
```

```
XPath Query#1: /PurchaseOrder/Order[2]/child::*

Resulting Node Set#1:
====================
    <Product Name="Adjustable Worktable" Price="3000.00"
     Quantity="1"/>
```

XML Style Sheet Processing



Example of document transformation using XSLT



```
<PurchaseOrder>
  <Name> Plastic Products </Name>
  <billingAddress>
      <Number> 158 </Number>
      <Street> Edward st. </Street>
      <State> QLD </State>
      <PostalCode> 4000 </PostalCode>
      <Country> Australia </country>
  </billingAddress>
<PurchaseOrder>
```

```
<PurchaseOrder>
<Name> Plastic Products </Name>
  <billingAddress>
     <Number> 158 </Number>
      <Street> Edward st. </Street>
      <PostalCode> QLD 4000 </PostalCode>
      <Country> Australia </country>
  </billingAddress>
<PurchaseOrder>
```

Example of document transformation