2 Handling Semantics in Information Exchange – An Introduction to XML

Good information is hard to get.
It is even more difficult, to use it.
[Sir Arthur Conan Doyle (1859 – 1930)]

This citation is even more valid for the information age with many billions of Web page and in these times of the semantic web. The main issue for Web users is not getting information. Rather the principal task for the user of the Web is to separate relevant from irrelevant information. Humans can use their knowledge and subtle clues when they face a long list of results from a search engine. Conversely, when there is automated processing of information, large numbers of unreliable search results are often a serious problem.

Opening Vignette: Will XML be the ultimate platform? Or will it be the next EDI?[1]

I hear it's going to cure cancer," says Tim Bray, its co-creator. "It's going to do my dishes, I hear," says Anne-Marie Keane, Staples' vice president of B2B e-commerce. Behind the flip jokes lies XML - a syntax that underpins a growing list of more than 300 nascent data standards. MathML, for instance, will make it possible to manipulate advanced mathematical equations on a Web page. Spacecraft Markup Language standardizes databases that operate telemetry and mission control. And then there's MeatXML, a comical name for a serious effort to create a universal meat and poultry supply chain standard. With XML going in so many directions at once, you can't blame CIOs for being confused. The hyperbole often makes XML sound like a salve for all pain. Finding the truth behind the tales takes some digging. Technologically, XML is a giant leap for IT. It can drastically reduce development time while making data transfer over the Internet simple. If nurtured properly, it may even become the ASCII text of online business - ubiquitous and assumed. Or it could become the next EDI, fractured under the pressure of vendor self-interest. One thing is certain: for XML to reach its full potential, CIOs will have to take an active role in forcing their partners, their vendors and even their competitors toward a radically more open computing model than what existed before.

This chapter deals with enhancing the understanding of an electronic document and information exchange. The eXtensible Markup Language (XML) and

[1] Excerpted from Scott Berinato, "XML: The Hype Stuff", CIO Magazine, available at http://www.cio.com.au/index.php/id;212673785.

derivatives are nothing other than an approach to realize the understanding of an electronic document. XML is a foundation of the semantic web. In recent years, discussion is not just on markup languages. Increasingly the dialog about exchange of information includes also the term ontology. This term is rather scary for a number of people who are not intimately involved in the field. The task of ontologies is merely to describe things and the relationships between things. This increased emphasis on ontologies has come into focus by the semantic web.

Over many years, organizations and public agencies have come forward to develop standards to solve communication hurdles. These standards exist to establish and disseminate communication methods to allow smooth understanding in information flows. For example, SWIFT is an important communication standard that supports automated money transfer between banks. Some communication standards are already accepted. For example, the SWIFT codes to facilitate money transfers are encoded in an International Standards Organisation (ISO) standard (ISO 9362). There is even an XML wrapper for SWIFT codes (swiftML). There are also emerging or *de facto* standards. The XML recommendations from the World Wide Web Consortium (W3C) are good examples of such *de facto* standards. They only have the force of the consortium behind them. Yet a wide array of important standards rest on these *de facto* standards.

XML is a basis for several communication standards. The reasons for development of XML were many problems with the HyperText Markup Language (HTML). These issues are seen in the so-called HTML dilemma. To understand this dilemma, the following Code example 2.1 shows the structure of a typical HTML document.

```
<html>
     <head>
     <title >Skeleton of a HTML document</title>
     </head>
     <body>
     <h1>HTML documents</h1>
     <p>This is a simple <strong>HTML document
     </strong>without navigation<p>.
     </body>
</html>
```

Code example 2.1: Document structure using HTML

HTML is a markup language for the standardized display of information. Web browsers take tags such as <h1> (highest heading level) and <p> (paragraph) as clues to represent information on the screen for a human user. Different browsers have a different understanding about the document structure information. Seeking a more positive user experience, websites will often use HTML tags for other than their intended purposes. As a result the content (or better the semantics) of a HTML document remains unclear for information exchanges purposes.

The dilemma of HTML is that it is on the one hand very easy to structure documents for Web presentations - which is positive. On the other hand, HTML does not allow a machine understandable structuring of documents. Search engines such as Google, allows ready access to a wide variety of documents on the Internet. Unfortunately, that variety of search results is often a major impediment to finding a desired value. All search engine users have the experience of a result list with a million or more web pages. Most of these pages are not relevant for the user. The result list is at the level of the relevance of the complete content in each document, which is inappropriate in the realm of intra- or inter-enterprise communications of designated values such as *Revenue* or *Qualified Carbon Credit*. Automated document retrieval by search engine cannot ensure document or value validation - which is negative. The construction of XML plays an important role in resolution of these existing problems.

2.1 What is XML?

The eXtensible Markup Language (XML) is a World Wide Web Consortium (W3C) defined open standard. Its task is to contain, name, structure, and secure information. Thereby XML is not a programming language. The task of XML is to describe the content of an electronic document. This means to clarify the meaning of *name*, *address* and so on and then to structure it in an appropriate format (the first row of a document is *name*, second row is *street* etc.). XML is a text-based meta-language to exchange, visualize and manipulate structured data to support heterogeneous applications.

The following example (Fig. 2.1) shows textual information, typed in a text-processing program such as Microsoft Word. A human reader can read and understand the text, which is nothing else than first name and surname. If a user wishes to store this information, the application stores it as a single document with a number of characters without knowing the meaning of these characters. XML manages information in an entirely different way. First, XML defines the structure of a document. Second, XML fills the information structure with appropriate values. This leads to human- and machine-readable documents. XML has a similar appearance to those structured in the HTML. HTML is only able to define the presentation of the stored information. XML defines content. As a result, XML is an important foundation of the semantic web.

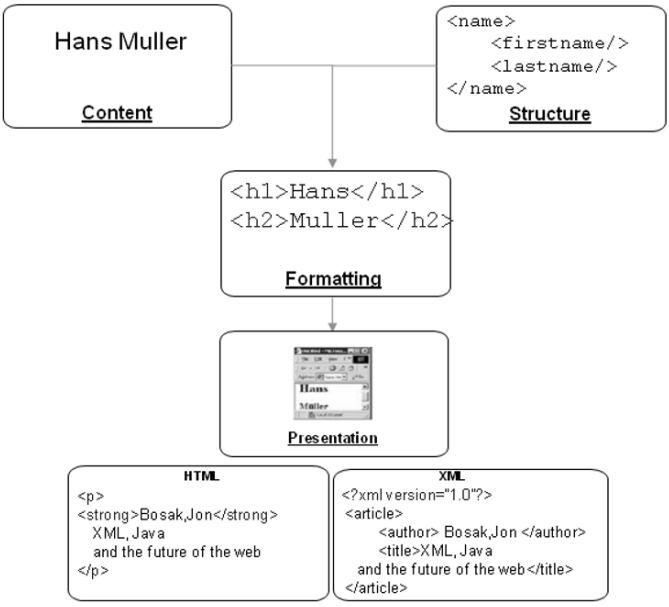

Fig. 2.1: XML task, HTML and XML

Importantly, XML is an open standard. It is an information container and a toolset for other more specific markup languages, such as XBRL. XML documents are machine- and human-readable, although the level of readability varies with the level of their complexity. They are also structured like a tree, which is a very common pattern in computer science. The following ten aspects characterize XML:

1. XML represent structured data.
2. XML looks a bit like HTML.
3. XML is text, but not for reading.
4. XML has a full design.
5. XML is a family of techniques[2].
6. XML is new, but not that new.
7. XML transfers HTML to XHTML.
8. XML is modular.
9. XML is the basis for the Semantic Web.
10. XML is freely licensed, platform- and company independent and well supported.

Besides the structuring of documents, data can be stored and retrieved by using XML standards and tools. Additionally, the output can be formatted by using

[2] XML is a collection of a number of languages, rule collections and data structures. It contains e.g. XSLT, XPath, XLink and XML schema.

XML together with style sheets (CSS). They can be transformed into an HTML document (XML and XSLT), a PDF document (XML and XSL-FO) and many other formats. As already stated, XML describes data. The following Code example 2.2 shows an email described by XML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<email>
    <from>jenny</from>
    <to>benny</to>
    <subject>Hi</subject>
    <message>Hello Benny! How are you doing?</message>
</email>
```

Code example 2.2: Email as an XML file

This example is easy to understand, because the tags are readable and context oriented. There is no limit about the complexity or the hierarchy depth of an XML document. The logical structure works in a hierarchical order, where the stored information is represented as concepts. As the concepts are interrelated, we model the interdependencies in a hierarchical tree structure. As far as the physical data structure is concerned, the XML document is nothing more than a string of characters.

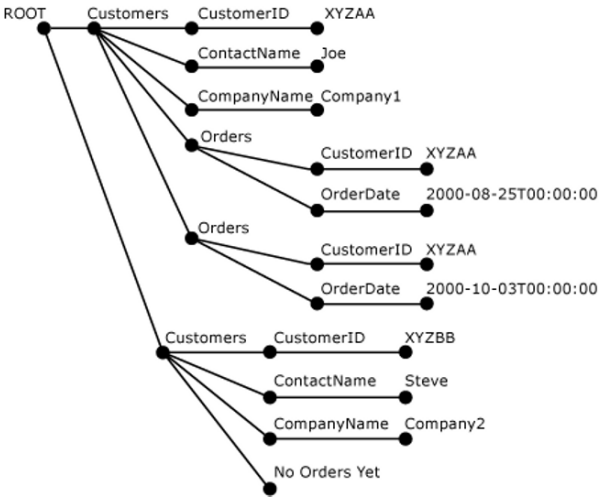

Fig. 2.2: Tree of an XML document

XML is a standardized language that includes the capability to define the markup language syntax. Since the definition of XML in 1998, many more initiatives have come from the XML community. The following shows the most important XML language family members.

- **XLink** (Recommendation, 27 June 2001)
- **XPointer** (Recommendation, 25 March 2003)
- **XML Namespaces** (Recommendation, 14 January 1999)
- **XSL** (Recommendation 15 October 2001)
- **XSLT 1.0** (Recommendation, 16 November 1999),
- **XSLT 2.0** (Recommendation, 23 January 2007)
- **XPath 1.0** (Recommendation, 11 November 1999),
- **XPath 2.0** (Recommendation, 23 January 2007),
- **XQuery 1.0** (Recommendation, 23 January 2007).

XML document: The following figure shows the construction of an XML message.

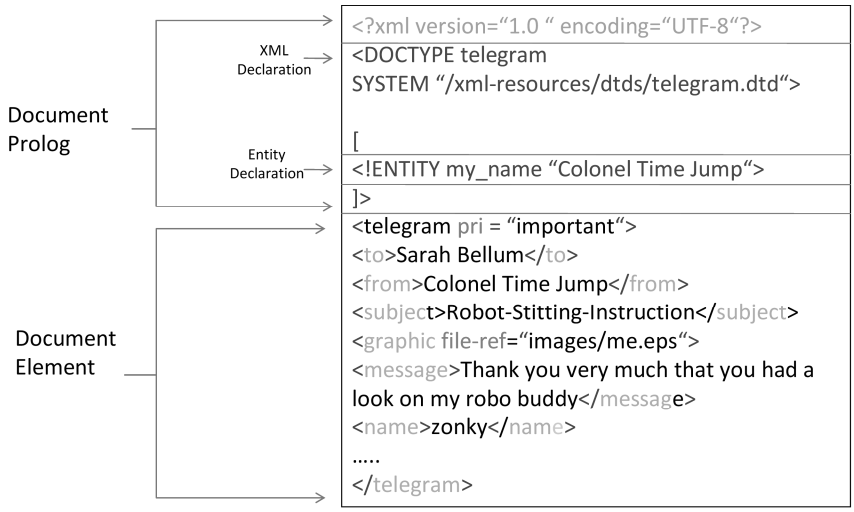

Fig. 2.3: Construction of an XML document

The XML declaration is a collection of information to prepare software processors for use of the information stored in an XML document. The declaration is always the first row of the XML document.

A container tag, or just tag or element, is a markup symbol which is symbolized by < > brackets. There must be a closing tag for each open tag. Tags symbolize that there is information about the semantics of the document structure. The following example shows the syntax of a container tag.

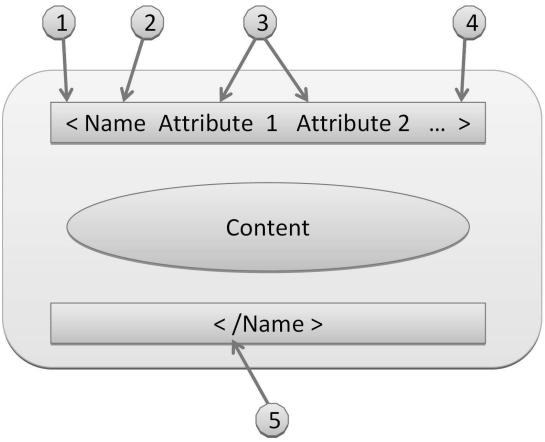

Fig. 2.4: Syntax of a container element

The figure above shows an (1) opening bracket, an (2) element name, (3) optional attributes of the initial tag, a (4) closing bracket and a (5) closing tag with forward slash after its opening bracket. The following syntax rules apply:

→ Tags cannot be overwritten (comparable to brackets in mathematic formulas)

```
Wrong:     <b>This text is bold <i>and italic</b></i>
Correct:   <b>This text is bold <i>and italic</i></b>
```

Code example 2.3: Example of nesting of tags

→ Capitalization matters (because the computer differentiates according to the case of text)

```
Wrong:     <Text>This text is bold and italic</text>
Correct:   <text>This text is bold and italic</text>
```

Code example 2.4: Example of capital case use in XML

→ Correct structure (each element can have subelements but they need to be properly nested as presented in the Examples 2.3 and 2.5)

```
<root>
  <child>
    <subchild>....</subchild>
  </child>
</root>
```

Code example 2.5: Example of proper nesting in XML

→ Valid characters for element names are 0-9, a-z, A-Z, dot (.), hyphen (-) and underline (_). The initial character of an XML name must be a letter. The element

name must not start with a hyphen, number, punctuation character or string *xml* (XML, Xml, xml, etc,). The length of an element name is not limited.

```
Correct    <_-_>allowed</_-_>
Wrong      <-_>not allowed</-_>
Wrong      <2nd-phone-number>13</2nd-phone-number>
Wrong      <advise+question>Bla</advise+question>
```

Code example 2.6: Example allowed and disallowed characters for tags

The content of an XML document comes through information within the tagged element, attributes of the tag, and sub-elements of the tag in an information structure. Attributes characterize things of the reality we want to describe in our document. Each attribute is as a container storing appropriate information. Technically, an attribute is a pair of a name and a value. For example by the means of attributes we can give an element a unique id. Attributes can provide support for differentiation of data, because the definition and amount of attributes is set up once and used several times. We show an example in the following figure.

```
<product id="screen-15-Zoll-Sony">
       <weight>...</weight>
       <resolution>...</resolution>
       <!—More product details-->
</product>

<product id="screen-15-Zoll-IBM">
       <weight>...</weight>
       <resolution>...</resolution>
       <!—More product details-->
</product>
```

Code example 2.7: Details of data contained in attributes

The following example shows rules for expressing values of an attribute.

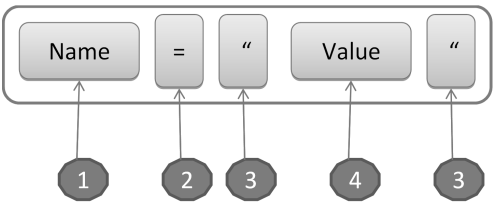

Fig. 2.5: Attribute example

The components of the Fig. 2.5 are (1) attribute name, (2) equal symbol, (3) double or single quotations, and (4) attribute value. The number of attributes of an element is not limited. The following Code example 2.8 shows a tag with four attributes.

```
<employee sex="female" age="25" height="170cm" ssn="123">
      <!—content of the tag-->
</employee>
```

Code example 2.8: Tag with four attributes

Attributes of the same element must have different names.

```
wrong: <team person="Jane" person="Joe"></team>
correct: <team person_1="Jane" person_2="Joe"></team>
```

Code example 2.9: Unifying attributes

In the previous examples, we demonstrated how data could be stored as value of an attribute. However, data can be also stored without use of attributes. The following example shows the same data stored as values of elements. In later chapters, we will explain how elements and attributes are used to capture certain financial reporting data.

```
<team>
      <person1>Jane</person1>
      <person2>Joe</person2>
</team>
```

Code example 2.10: Content without using attributes

Attributes can be used to describe values. Other possibilities to get machine-readable semantics are elements. Attributes were introduced to describe information that is contained in the element itself as its value.

```
<from name="Maria Lara" email="ml@xyz.com">I am coming
soon!</from>
```

Code example 2.11: Information container

The final table summarizes the tag types in XML.

Tab. 2.1: Tag types in XML

Object	Reason	Example
Empty element	Represents information at a specific point within the document.	<xref linkend="abx"/>
Container element	Groups elements and character set data.	This is a new section. <!ENTITY Author “John Doe”>
Declaration	Enhances the parsing environment with new	<?print formatter create new row?>

	parameters, entities or grammar definitions.	
Comment	Enhances the document with a comment which is ignored by the XML processor.	<!--I stopped here-->
CDATA section	Creates a section with character set data that are not parsed (to keep all specific elements within the document).	<![CDATA[AND-CASCADE!&&&&&&]]>
Entity reference	Informs the parser to fill in textual parts that are stored at a different location.	&companyname;

2.2 XML namespaces

When we draw data from a variety of different sources, identical names can have different meanings. For example, the term *article* can mean an article of a book club, an article in a magazine or an article within the articles of incorporation of a company. To solve this problem, we can use namespaces coupled with element names to give truly unique combinations.

Fig. 2.6: Namespaces

Namespaces are parts of the XML family in general and of XBRL in particular. XML namespaces are a mechanism to link elements and attributes into groups.

We are going to take a closer look into XML applications that use attributes and elements within a single XML document (markup vocabulary). Such XML documents are used and understood by different software modules. It is more appropriate to reuse markup vocabulary as defined in the XML document. Such reuse of markup vocabulary is hindered when recognition and collision problems occur[3] in XML applications. Thus as a rule, tags and attributes must be unique. Collisions, where we have identical attribute names or element types, should not then cause errors during processing. XML namespaces exactly satisfy this requirement.

[3] The usage of identical terms with different meaning can lead to problems. For example, *Orange* can be a fruit or can also be a European mobile phone company.

Recommended XML Namespace for Government Organizations[4]

The US federal government is actively engaged in developing and deploying XML. It is critical that the government establishes a cohesive, coordinated namespace approach to support its various XML efforts. This namespace approach must define a standardized structure for federal namespace as well as establish a standardized naming convention for those namespaces. Without such a coordinated approach, individual government organizations will create a proliferation of disparate XML namespace structures and names resulting in chaotic management of XML components.

To summarize, we define the term namespace as follows: An XML namespace is a collection of names, identified by URI links that are in use as element types and attribute names within XML documents. URIs are necessary to be able to identify namespaces. To make it clear - URIs are part of the set of URLs. The syntax is of a namespace declaration is:

```
xmlns:prefix="URI"
```

The following example shows a code part of an XML document that uses namespaces to demonstrate the unique identification of attributes.

```
<order
   xmlns:bo="http://www.book.com/"
   xmlns="http://www.magazine.com/">
  <bo:article number="1234">
      <bo:description>XML and XBRL</bo:description>
  </bo:article>
  <article id="111">
      <name>Information Systems<name>
  </article>
</order>
```

Code example 2.12: Namespaces

The example contains namespaces with the attribute declaration *xmlns:bo* and *xmlns*[5]. The elements of the shown order belong to the first or the second category.

[4] Excerpted from J. L. Glace and M. R. Crawford: Recommended XML Namespace for Government Organizations, August 2003, available at http://xml.gov/documents/completed/lmi/GS301L1_namespace.pdf.

[5] *xmlns="http://www.magazine.com/"* is treated as default namespace and therefore is not assigned a prefix.

2.3 XML schema

The W3C developed XML schema as a language for the description of XML documents. An XML schema contains the necessary element types, attributes and entities to align with XML instance documents. XML schema is similar to a database schema for a database such as Microsoft Access or Oracle. An instance document contains information that aligns with the schema. The instance document includes processable information encoded by the tags from the schema. The XML schema defines syntactical rules and the structure of a class of XML documents.

A corresponding XML schema assures the validity of an XML instance document. XML parsers or validators will validate whether an instance document is a well formed XML document. Additionally the schema is an explicit coding of rules which contain information about what is valid and necessary and what is unnecessary.

Besides the already existing XML-based languages, XSD (XML schema description language) has the highest practical recognition. XBRL uses XSD as we describe later in the book. An XML schema can ensure a number of validation quality checks:

- It defines the elements that can be used within a document.
- It defines the attributes that can be used in a document.
- It defines the child elements of an element.
- It defines the order of child elements of an element.
- It defines the number of child elements.
- It defines if an empty element is allowed.
- It defines data types.
- It defines fixed and default values.

An XML schema supports differentiation of namespaces that go hand in hand with the structure of well-formed constraints on the content of an instance document.

An XSD document is organized as a tree of schema elements. The root of the schema file is always <*xsd:schema*>. The root element includes other elements <*xsd:complexType*> or <*xsd:simpleType*>. The prefix *xsd* shows that all elements belong to the XML schema family. The XML schema namespace declaration is:

`xmlns:xsd=http://www.w3.org/2001/XMLSchema`.

A typical XSD schema documents has several different components. Primary components are simple or complex type definitions, element or attribute declarations. We explain the roles of these components using a number of examples in this section. As shown in the following example, simple type definitions contain just a single value such as a number, a date or text. The XML schema means that an XML document created upon such a schema is supposed to provide a value for the element *name*.

```
<xsd:element name="name">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
</xsd:element>
```

Code example 2.13: Simple data type

Complex type definitions are elements that contain subelements or attributes. This means that an XML document based on the following schema should provide values for *projectMember*, *projectBudget* and *projectGoal*. All these three exist within the element *projectDescription*.

```
<xsd:element name="projectDescription">
<xsd:complexType>
  <xsd:sequence>
    <xsd:element ref="projectMember"/>
    <xsd:element ref="projectBudget"/>
    <xsd:element ref="projectGoal"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:element>
```

Code example 2.14: Complex data type

Secondary components are definitions of attribute groups, key references for uniqueness, declaration of comments and linking of names. The XML schema has additional supporting components like comment and description about using attributes.

We can use a *simple type* to express limitations such as *length, minLength, maxLength, pattern, enumeration* and *whiteSpace*. Let us look at the following examples:

```
<xsd:simpleType name="password">
  <xsd:restriction base="xsd:string">
    <xsd:minlength value="8"/>
    <xsd:maxlength value="15"/>
  </xsd:restriction>
</xsd:simpleType>
```

Code example 2.15: Password with minimum and maximum number of elements

An XML document based on the schema from Code example 2.15 can provide an alphanumeric value for the element *password* that has between 8 and 15 characters.

```
<xsd:simpleType name="dayOfWeek">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Monday"/>
    <xsd:enumeration value="Tuesday"/>
    <xsd:enumeration value="Wednesday"/>
    …
  </xsd:restriction>
</xsd:simpleType>
```

Code example 2.16: Enumeration of elements

An XML document based on the schema from Code example 2.16 can provide a value for the element *dayOfWeek* that is from the restricted set (*Monday*, *Tuesday*, *Wednesday*…). One element can hold just one of these enumerated values.

```
<xsd:simpleType name="isbn">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0-9]{10}"/>
  </xsd:restriction>
</xsd:simpleType>
```

Code example 2.17: Fixing an example for an element

An XML document based on the schema from Code example 2.17 must provide an alphanumeric value for the element *isbn* consisting of digits between 0 and 9.

```
<xsd:simpleType name="ageGroup">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="18"/>
    <xsd:maxInclusive value="30"/>
  </xsd:restriction>
</xsd:simpleType>
```

Code example 2.18: Defining a range

XML documents based on the schema from example 2.18 can provide a value for the *ageGroup* element that is an integer with the value between 18 and 30.

```
<xsd:simpleType name="direction">
  <xsd:list itemType="xsd:string"/>
</xsd:simpleType>

XML document (instance):
<direction>North East South West</direction>
```

Code example 2.19: Defining a list

The XSD definition of the document structure provides a set of names for all possible elements and their attributes as well as links them with their respective data type. We can add additional constraints and set default values of elements. We define the element declaration with the statement *<xsd:element name="description"*> (first character of a name is a letter). The XML schema allows us to define complex elements each of which have several child elements. We define this by modeling a custom data type:

```
<xsd:element name="description" type="description">
<xsd:complexType name="description">
   <xsd:sequence>
     <xsd:element name="customer" type="customer"/>
     <xsd:element name="position" type="position"/>
   </xsd:sequence>
</xsd:complexType>
```

Code example 2.20: Complex element

We can define complex data types with a so-called sequence. The term sequence[6] means that XML parsers will parse the lines in the instance document row by row.

```
<xsd:complexeType name="customer"">
   <xsd:element name="name" type="xsd:string"/>
   <xsd:element name="street" type="xsd:string"/>
   <xsd:element name="postcode" type="xsd:string"/>
   <xsd:element name="place" type="xsd:string"/>
</xsd:complexeType>

Additional fixed value:

<xsd:attribute name="currency" type="xsd:string"
fixed="Euro"></xsd:attribute>
```

Code example 2.21: Complex element defined by sequence

2.4 XLink

This section briefly explains XLink, the XML linking language. XLink is as important to XBRL as is XML namespaces. The main goal of XLink is to enable linking within and between XML documents. In order to provide this functionality, XLink allows the creation of unidirectional links as well as a complex link structures. Links in XML documents can bind elements with

[6] The sequence element specifies that the child elements must appear in a sequence. Each child element can occur from 0 to any number of times. Parsers themselves unlike compilers traverse the code line by line (of any XML document or XML schema document).

individual or multiple resources[7] or with associated metadata[8]. Resources in XLink can be external websites but also other documents or sections of documents. This means that XLink is nothing else than a set of destinations (so-called resources) and a set of links between these destinations (the solution for creating links in XML documents was to put a marker on elements that should act as hyperlinks). A link contains all necessary information, to be able to decide which appropriate resources have to be linked. This does not mean that all destinations in use in a link must have a relation to another destination.

```
<?xml version="1.0"?>
<homepages xmlns:xlink="http://www.springer.com">
 <homepage xlink:type="simple"
 xlink:href="http://www.springer.com">Visit
Springer</homepage>
 <homepage xlink:type="simple"
 xlink:href="http://www.iasb.org">Visit IASB</homepage>
</homepages>
```

Code example 2.22: Linking with XLink

This example presents a way by which simple hyperlinks can exist in XML documents.

XLink has two parts: *simple* and *extended links*. *Simple links* are a connection between two resources. This is comparable to the links known in HTML. Figure 2.7 shows a *simple link* to connect a local and a remote resource. This can be for example terms in a text document that link to their definition.

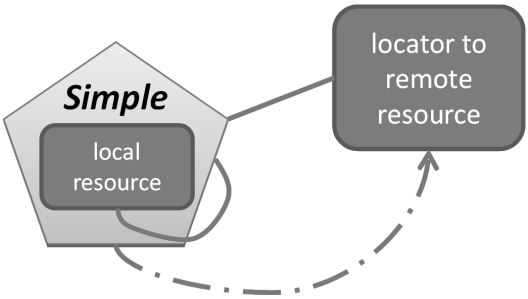

Fig. 2.7: Simple link

A *simple link* is a limited option to connect two resources. The *extended link* offers the full XLink functionalities. The *extended link* allows the association to an

[7] A resource can be everything which has an own identity. The term resource has in context to the World Wide Web (WWW) different meanings. It can be an image, a file, a program or another XML document.

[8] A link can contain additional semantic information.

open number of resources. The participating resources can be a free combination of local and remote destinations. The following figure 2.8 shows an example of the *extended link*.

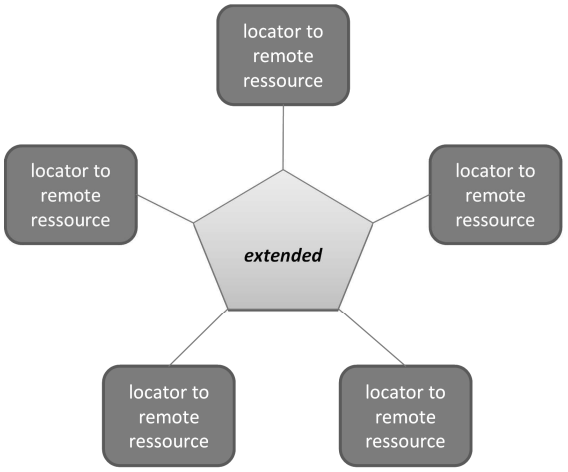

Fig. 2.8: Extended link

XLink

A language designed to allow creation of hyperlinks in XML documents. Hyperlinks may have a specific meaning (role). Using XLink allows for connecting elements and describes the nature of the connection in a computer-readable manner. XLink hyperlinks connect internal (within one XML document) and external resources (resources within the document to external and even non-XML resources). In the XBRL architecture, XLink defines the various linkbases, including presentation, calculation, definition, label and reference linkbases by using the arcs and locator methodology. The arcs can be given a specific meaning to indicate the nature of the relationship. For instance, arcs in the XBRL presentation linkbase are defined as a parent-child. This indicates the hierarchical meaning of the relationship. Arcs in the calculation linkbase are defined as summation-item indicating the aggregation nature of the relationship.

2.5 Summary

- XML is a World Wide Web Consortium (W3C) defined open standard. Its task is to contain, name, structure, and secure information. XML and derivatives are nothing other than an approach to realize the understanding of an electronic document.
- XML defines the structure of a document and fills the information structure with appropriate values. This leads to human- and machine-readable documents.
- Content of an XML-document comes through information within the tagged element, attributes of the tag, and sub-elements of the tag in an information structure.
- An XML namespace is a collection of names, identified by URI links which are used as element types and attribute names within XML documents.
- The XML schema defines syntactical rules and the structure of a class of XML documents.
- The main goal of XLink is to enable linking within and between XML documents. In order to provide this functionality, XLink allows the creation of unidirectional links as well as a complex link structure.

2.6 Key terms you should know

- XML
- XML documents
- Tags
- Elements
- Attributes
- XML namespaces
- XML schema
- XLink

2.7 Case analysis[9]

Improving communications between departments was one of the main reasons why governments around the world began showing early interest in XML, which emerged in 1998 as an official World Wide Web Consortium standard that built on the runaway success of its smaller cousin, the HyperText Markup Language (HTML).

Integration into all manner of products came quickly, and XML's ability to add structure and meaning to data made it a philosophical companion to efforts to open up government data storage formats. Equally popular was XML's enablement of industry or function-specific markup schemas – eXtensible Business Reporting Language (XBRL) is one, as are eBusiness XML (ebXML), Financial Products Markup Language (FpML), Math Markup Language (MathML) and myriad others with specific purposes.

Yet XML's flexibility may have muddied its message: even the official World Wide Web Consortium family of XML-based standards has grown, now including XPath, XPointer, XSLT and a host of others. With so many acronyms floating around - and specific expertise needed to take advantage of each one - it is no wonder that many organizations that might otherwise be interested in XML's benefits have put it into the too-hard basket.

Just because workers can create XML documents does not mean they will. XML is an enabler, not a destination, and will offer no real benefit without appropriate policy and framework formulation.

[9] Excerpted from David Braue, “You Can Lead a Government to XML . . .”, CIO Magazine available at http://www.cio.com.au/index.php/id;1160512555;fp;4;fpid;21.

http://www.springer.com/978-3-642-01436-9

XBRL for Interactive Data
Engineering the Information Value Chain
Debreceny, R.; Felden, C.; Ochocki, B.; Piechocki, M.;
Piechocki, M.
2009, XXXII, 214 p., Hardcover
ISBN: 978-3-642-01436-9