1. INTRODUCTION

A problem of business documents transfer in 95% [1] is reduced to exchange some standard information “packages” such as products’ offer, purchase order, despatch advice, request for payment, invoice, etc. The traditional way of information exchange (phones, fax, post, visits of sales representative) is expensive and time-consuming, takes a significant firm’s resources (people, money, time).

The Internet development and its common accessibility create conditions, in which business activity gets a new quality – electronic business. As main advantages of network using in business processes are listed speed and a low cost of data exchange. It was became cause of integration of the organisations computer systems, what was the first step to create Electronic Data Interchange (EDI) standard.

2. EDI - THE RECOGNISED STANDARD

Electronic Data Interchange technologies have evolved as data carrier replacing the paper documents more than two decades ago. The first electronic interchanges used proprietary formats agreed upon between two trading partners, and it required a new program every time when a new partner was added to the system. Next some industry groups started a co-operative effort to develop industry EDI standards for purchasing, transportation, and financial applications. Many of these standards supported only intra-industry trading (based on Intranet), what resulted a large number of EDI formats.

In 1979, the Accredited Standards Committee (ASC) X12 was constituted to develop a generic EDI standard. In 1993, Version 3, Release 4 contained 192 standards [6]. There are over 100 additional standards in development. In the U.S., the most commonly used standard is ANSI X12, co-ordinated by the American National Standards Institute (ANSI)[12]. While in Europe, it is the Electronic Data Interchange for Administration, Commerce, and Transportation (EDIFACT)[18] standard.

Now Electronic Data Interchange standards are very known and recognised, every institution, which is looking for savings (money and time), would like to have it. But problem of EDI technology is that it is too complex. It requires special proprietary software and powerful hardware (communication servers). This way EDI technology is prohibitively expensive to implement for small and medium-sized companies. For them the XML applications will be a good resolution.

3. OVERVIEW OF XML TECHNOLOGY

XML (eXtensible Markup Language) is relatively young, but very fast and still developing technology. The first time XML was presented at the SGML 96 Conference in Boston (November 1996). The 1st XML Conference was held by the Graphic Communications Association (San Diego, March 97). In October 97 the W3C
came with a note on “W3C Data Formats on XML, SGML, HTML, and RDF”[21].

Until now the XML specification is backed by SoftQuad, Adobe, IBM, HP, Microsoft, Netscape, Lockheed Martin, NCSA, Novell, Sun, Boston University, Oxford University, and the Universities of Illinois and Waterloo [4].

XML is a meta-markup language that provides a format for describing structured data. “Meta” – it means that XML allows to create others customised markup applications e.g.

- MathML – Mathematical Markup Language [9],
- SML – Synchronised Multimedia Integration Language [10],
- VML – Vector Markup Language [11],
- HRMML – Human Resource Management Markup Language[13],
- IFX – Interactive Financial eXchange [14],
- SIF – Schools Interoperability Framework [15],
- CBL – Common Business Library [16],
- ebXML – Electronic Business using eXtensible Markup Language CBL [17],
- PDML – Product Data Markup Language [19],

Those data viewing and manipulation applications are dedicated for groups of people and organisations specialised in one kind of activity.

XML has much in common with data description languages, such as DDS and SQL, but there is a major conceptual difference. An XML document is completely self-contained and includes both data and data description. Other languages only describe data (field names, data types, and sometimes—valid field values). The data itself are sorted in relational database independent of the data description. With XML, a programmer should be able to extract the meaning of the data from a document without additional information.

On the surface, XML looks like HTML (Hyper Text Markup Language). Both are derived from the Standard Generalised Markup Language (SGML)[20]. The tools that generate HTML can often be reused to generate XML[8]. Both HTML and XML use marks <, > and & to create element and attribute structures.

The main difference is that XML hasn’t got a limitation of pre-set library of tags. When the document is marked up in XML, it is possible to choose the tag name that is the best way describes the contents of the element.

3.1. XML Syntax

The XML documents must follow rules for identifying document parts and creating nested element structures. The elements identify named sections of information and are built using markup tags that identify the name, start, and end of the element. The elements in XML documents can not overlap. If the start tag for an element appears within another element, it must end within the same containing element.

XML is similarly strict about other aspects of syntax [8]. All attribute values must be quoted, whether or not they contain spaces. Option of single- or double-quotes still exists. The characters <, > or & cannot be used within the text of documents. They are replaced by the built-in entities &lt;, &gt;, or &amp;.

One element can represent one record of some data, where each property is defined by attributes of the element (see
An attribute allows for adding information about an element using name-value pairs. An element may not have two attributes with the same name. The following abstract sample document illustrates the relationships between elements.

![Correct and incorrect XML syntax](image)

**Fig. 1.** Syntax of relationships between elements

3.2. XML Semantics

Although XML is unforgiving about syntax, it offers the developers more options for the defining meaning in XML documents. HTML is basically one vocabulary with a few variations; `<b>` always means the same thing to an HTML processor. In XML it is possible to create as much markup words as it is necessary. This flexibility of XML language is also configurable by schemas, document type definitions and name space mechanisms [21]:

- Data Type Definition (DTD) is a series definitions for element types, attributes, entities and notation [5, 2] which defines the permissible tagging structure for a document. DTD is defined by its predefined elements and rules may be very straight or flexible. The XML documents do not required DTDs, they are borrowed from SGML. DTDs may be defined as a part of document or may be externally referenced.

- XML schemas are more flexible than DTD and probably will ultimately replace them. The schemas are developed specially to give the XML user ability of validates his documents. The schema offers the same ability of defines tagging structure as DTD, but more here it is possible to declare types of data. Here date is #DATE, money are #NUMBER, etc. - in DTD every data is #PCDATA. More schemas are full compatible with namespaces, DTD are not.

- Namespace are a concept introduced into XML to allow a more modular design. With their help data processing software can easily resolve naming conflicts in XML documents. Namespaces are declared in the same way as attributes; strictly speaking they are attributes. To make for example `http://pluton.pol.lublin.pl` the document's default XML namespace - in overhead should be written the predefined element `xmlns` then used name of prefix and address URL of namespace location (see figure 2). It is possible to use more than one namespace in one document.

The following example of business document (an invoice) illustrates the simplicity of XML code. Names of elements are humanly readable. Prefix `doc:` in front of element
name points to namespace http://pluton.pol.lublin.pl use and meaning of each element is read in context of it.

```xml
<?xml version="1.0" encoding="UTF-8"?>
< xmlns:doc="http://pluton.pol.lublin.pl/">
  <doc:Invoice >
    <doc:InvoiceHeader InvoiceNum = "I0302711"
                        InvoiceDate = "2003-08-05"
                        Currency="EUR">
      <doc:Party PartyID = "7013214165870"
                 Name = "University of Technology"
                 Street = "Nadbystrzycka 38a">
        <doc:PostalInfo City = "Lublin"
                         PostalCode = "20501"
                         Country = "Poland"/>
      <doc:Ref>PL1234567890</doc:Ref>
    </doc:Party>
    </doc:InvoiceHeader>
    <doc:InvoiceDetails>
      <doc:LineItem Num = "1"
                    ItemCode = "100444a"
                    ItemDesc = "Pasta Cmelloni"
                    Quantity = "100">
        <doc:ItemDetail UnitPrice = "0.55"
                        TaxCategory = "S"
                        TaxPercent = "15.00"
                        Date = "2003-08-04T15:00:54"/>
      </doc:LineItem>
    </doc:InvoiceDetails>
    <doc:InvoiceSummary>
      <doc:TaxSummary>
        <doc:Tax>
          <doc:TaxCategory stdValue="S"/>
          <doc:TaxPercent>15.00</doc:TaxPercent>
          <doc:TaxableAmount>55</doc:TaxableAmount>
          <doc:TaxAmount>8.25</doc:TaxAmount>
        </doc:Tax>
      </doc:TaxSummary>
      <doc:InvoiceTotals>
        <doc:NetValue>55.00</doc:NetValue>
        <doc:TaxValue>8.25</doc:TaxValue>
        <doc:GrossValue>63.25</doc:GrossValue>
      </doc:InvoiceTotals>
    </doc:InvoiceSummary>
  </doc:Invoice>
</doc:Invoice>
```

Fig. 2
Sample of invoice document in XML
3.3 Parsers

The parser is responsible for handling XML syntax and, if desired, checking the contents of the document against constraints established in a document type definition (DTD) or schema; the application must understand how to process or display the information. The application is isolated from the details of the XML document, allowing the document creators to take advantage of those details without worrying about the application. Document creators must still present correctly structured information to an application, but they can do so using generic tools instead of creating new interfaces for every different transaction. That makes the XML technology easier, cheaper and faster to programmers than, for instance, some applications based on *.txt files.

4. SECURITY

XML technology has no included secure mechanism. Now it is necessary to use some external solutions like Transfer Layer Security (TLS) [4] for transfer through a TCP/IP protocol or Secure Socket Layer (SSL) [4] when Web-servers are used. High degrees of an XML document structure helps to add some digital dedicated signatures to individual parts of a document as well as to the whole one.

5. BENEFITS OF XML

XML provides a powerful, flexible format for expressing data—whether as a wire format for sending data between client and server, a transfer format for sharing data between applications. Simple data describing allows programmers to export information from ERP systems and interchange it with other's back-end systems (platforms, operating systems) of business partners. XML is very programmable – there are a lot of ready parsers, which makes problem of analysing data XML format much easier than e.g. TXT files.

XML offers the following benefits from the business point of view:

- Standardisation,
- Manageability,
- Business-to-Business Communication,
- Accessibility,
- Portability,
- Time saving,
- Lower cost.

Ability of extensibility allows XML to describe each economy event. You can always build your own vocabulary. The advantage of data being independent of any particular platform, application or vendor, is that it can be transformed to produce different types of outputs for different media devices (Web browser, paper, CD-ROM) without the need to modify the original content. XML affords customisation and personalisation of information for greater user satisfaction.

XML application could offer functionality of EDI but for much lower costs. Even more, because of ability of XML human readable format, in some conditions it could
be used to exchange business documents with partner whose hasn’t got an ERP system (computer PC will enough). XML technology is flexible enough to handle an incredibly wide variety of information, and also allows for such information to be self-describing, so that it may be manipulated by software that has not been previously exposed to a description of the underlying meaning behind the data.

6. DISADVANTAGES OF XML

The main disadvantage of the XML technology is redundancy of data, because data in the XML document have its own unique address. The problem of redundancy does not exist in relational database.

Another problem is involved with international exchange of documents, because of the language, which is used to describe tags of elements and their attributes. The usage one language for the whole system could be a barrier for non-native end users. But using some tables of translation could solve that problem. The 'Tower of Babel' effect could be a real dangerous because of many local customised implementation of XML. But looking at position of ebXML application (it should become the standard soon), this problem will be gradually minimised.

REFFERENCES


WWW sites*
[10] http://www.w3.org/AudioVideo

* All links to the sites are actual on May 25, 2003