Electronic Commerce: The End of the Beginning

13th International Bled Electronic Commerce Conference

Bled, Slovenia, June 19-21, 2000

XML vs. UN/EDIFACT or Flexibility vs. Standardisation

Christian Huemer

Institute for Computer Science and Business Informatics
University of Vienna
Liebiggasse 4/3-4, 1010 Vienna Austria
Tel.: +43-1-4277-38443, Fax: +43-1-4277-38449
christian.huemer@univie.ac.at

Christian Huemer

Abstract

XML, the eXtensible Markup Language, has become the standard for defining data interchange formats in Internet applications. Therefore, it is currently one of the most popular topics in the area of Electronic Commerce. The XML-hype also enters the field of electronic data interchange (EDI). In the past decades EDI standards, like UN/EDIFACT or ANSI X12 have been the dominant ways of interchanging data between applications. These traditional standards are successfully used by the Fortune 1000 companies, but were not commonly accepted by most of the SMEs. Owing to its flexibility, XML is expected to close this gap. But there is a huge uncertainty among companies. Some are concerned that XML is a threat to their current EDI applications. Others are making technically naive and overly optimistic statements on how XML will replace current EDI standards. Both expectations are not entirely true. In this paper we describe the strengths as well as the limitations of using XML in EDI. By comparing XML with current EDI standard technology, we show where XML still has to learn from EDI standardisation.

1. Introduction

Electronic Data Interchange (EDI) is the application-to-application exchange of electronic business-related data based on a format understood by both (all) trading partners using an electronic transmission medium in order to carry out a business transaction [1, 6, 7]. Although the economic advantages of EDI are widely recognised, the number of organisations and companies employing EDI is relatively small compared

to the total number of businesses worldwide. This is due to the fact that the costs for setting up and running an EDI relationship are too high for SMEs. Many people blame the complexity included in current EDI standards, like UN/CEFACT or ANSI X12, for the limited success of EDI.

In contrast to EDI, the World Wide Web (WWW) has become a successful platform for conducting business electronically. End-consumers as well as SMEs find it convenient to perform business transactions by completing an electronic form, which can be automatically processed by the business partner. In the early days of the WWW, the information exchange was most commonly based on the common gateway interface (CGI) which allows the structured transfer of the content of an HTML form to the server where it is processed by a scripting language. Nevertheless, HTML based electronic commerce includes some shortcomings [2]: It is always necessary to load an entire page. This is not comfortable in cases where a stable template remains and only portions (e.g. resulting from a dynamic query) have to be changed. Furthermore, the quality of WWW searches could be improved. For example, to specify search criteria for a selected price range is effectively impossible, because there is no way to mark a string of digits as price.

The eXtensible Markup Language (XML) is designed to overcome the above mentioned shortcomings of a pure HTML approach [11]. As opposed to HTML tags referring to the presentation of data XML tags are used to describe what the information is about. Accordingly, XML makes it possible to encode information with meaningful structure and semantics in a notation that is both human-readable and processable by computers. Due to the fact that XML files could (under given circumstances) be automatically processed by computers, XML is proposed as an alternative for the application-to-application exchange of data [14].

Over the last two years many XML-based applications appeared. First success stories, like that of RosettaNet, underpin the XML strengths in the EDI area. The enthusiasm for XML created unrealistic public expectations about XML's potential to change the world. But XML, while quite powerful in terms of generality and flexibility, is not by itself a solution to any modelling or computing problem [4]. XML's flexibility provides for a fast and unbureaucratic way of defining XML-based interchange formats between business partners. However, these interchange formats are limited to a closed user community, and are most likely to be incompatible to others.

It must be realised that XML has to offer a lot of goodies for EDI, but employing XML alone does not guarantee for a plug-and-play solution in application-to-application information exchange. But it was the „easy-to-use" approach which made up the success of the Internet. A similar success in EDI could only be reached, if there were no need for an agreement on the meaning of XML tags (cf. Open-edi [10]).

Thus, we analyse the areas where XML is superior to current EDI standards. Furthermore, we show the current limitations of XML-based approaches and where these approaches can learn from EDI standardisation to help realising the „Open-edi" vision [9] in which companies could exchange messages to conduct electronic business in a completely ad-hoc fashion, with minimal prior agreements. However, it must be recognised, that the pure combination of XML and EDI strengths will not lead to an Open-edi environment, because XML is nothing more than a syntax that could be used in the functional service view (FSV). But using the enthusiasm about XML to design carefully the message flows in dialogues between applications on the basis of the business needs (business operational view) before transforming them into XML syntax will be a step into the right direction.

The remainder of the paper is structured as follows: In Section 2 we give an overview of the approaches to structure information in an interchange file for both UN/EDIFACT as representative of traditional EDI standards and XML. The advantages and disadvantages of using XML for EDI are presented in Section 3. Section 4 presents requirements to be met in the future in order to enable a global data interchange based on XML. Finally, Section 5 concludes with a short summary.

2. Approaches to Information Structures

In order to exchange data between applications, it is essential that the applications know what the data is all about. A data format specifying how data are to be represented ensures the identification of data values. In addition, the meaning of the data values must be defined to exchange electronic data across organisational boundaries in such a way that no human intervention for its interpretation is required. EDI standards must cover both syntax and semantics. Over the past decades various EDI standards have been developed: Proprietary formats, sector specific solutions and branch-independent standards like X12 and UN/EDIFACT. But all these „traditional“ standards use the common concept of implicit data identification, which differs from the tagging mechanism used in XML. The following two subsections present the different methods of ensuring the correct interpretation of information by information systems.

Current EDI Standards (UN/EDIFACT)

Since most of the current EDI standards use a similar method to ensure machine interpretation and describing all of them would involve a lot of redundancy, we have selected UN/EDIFACT as representative for the current EDI standards. UN/EDIFACT is designed to be an international and branch-independent standard. Its syntax has been adopted as ISO standard 9735 in 1987. In 1988 the UN/ECE agreed to maintain the UN/EDIFACT message types.

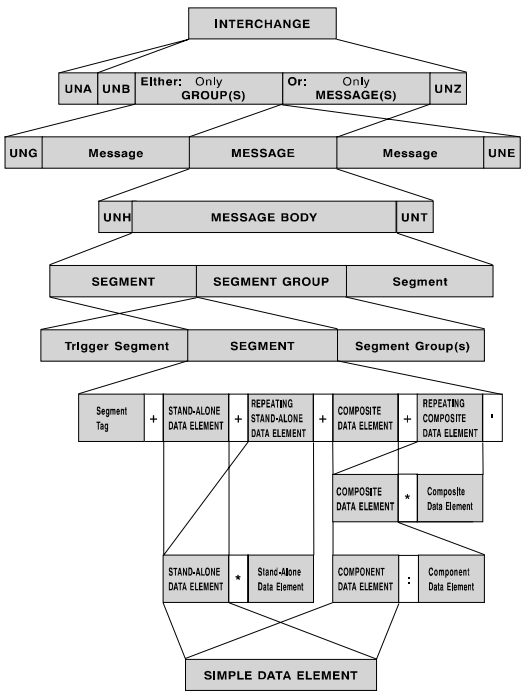

Figure 1: UN/EDIFACT Structure

UN/ EDIFACT is based on the following key concepts: Messages, Segments, Data Elements and Codes. Standardised Codes are used for representation of business terms. Data Elements are the smallest indivisible pieces of data. Furthermore, UN/EDIFACT uses the concept of composite data elements, which are sequences of simple data elements that all together describe one logical unit. Segments are groups of related data elements. A message is a sequence of segments and segment groups representing a specific business transaction. This hierarchical

structure which is used in an UN/EDIFACT interchange is depicted in Figure 1.

Taking a closer look at Figure 1, it is easy to detect that UN/EDIFACT takes advantage of a delimiter-based syntax: The data values are separated from each other by special characters („:“, „+“, etc.). This allows for flexible length of data values.

Of great significance is the fact that UN/EDIFACT is based on implicit data identification. Accordingly, the meta information (the meaning of a data value) is **not** explicitly stated in an interchange. The semantics of a data value are given by its position in the message. Consequently, simple data elements within a composite data element, data elements within a segment, as well as segments in segment groups and messages must follow a predefined order. This order is defined in the message type definitions that are maintained by the UN/ECE and published twice a year in UN/EDIFACT directories. To ensure a common understanding of transmitted data values both business partners must be aware of the corresponding message type definition.

Although the right interpretation of transmitted values is the main function that an EDI standard must ensure, EDI standards cover also the following two topics: business knowledge caption and agreed code lists. Each UN/EDIFACT message type is a data model for a business transaction. It is created by volunteers from the business world who put their business sector know-how into a data model which is written down in UN/EDIFACT notation [15]. As a result an UN/EDIFACT message type is a data model that is intended to capture all semantics that may appear in a corresponding business transaction type. But the careful analysis to create an UN/EDIFACT standard message type also results in a serious drawback, because the standardisation process is time-consuming. The usage of agreed code lists is also crucial for the UN/EDIFACT approach, since free text statements are not well suited for further processing by a machine.

XML

XML is a W3C standard since February 1998, but the development of XML started already in 1996. Nevertheless, XML is not a new technology. In fact, XML is a subset of the ISO standard 8879 SGML (Standard Generalized Markup Language), which was developed in the beginning of the 80´s and approved in 1986. SGML has been successfully used in large applications of publishing companies. But SGML is quite complex and includes many features that are not of first priority in Internet applications. Therefore, XML has been designed to use 20% of SGML´s features to get 80% of SGML´s functionality.

XML is a markup language using tags and attributes to mark data. But XML is not a screen-formatting language like HTML. XML itself does not even care about presentation. XML is designed for more intelligent applications where the meaning of data and not (only) its presentation is crucial –, as it is the case for EDI.

XML is a language to describe structured data in a way that the information is self-explaining. Therefore, XML is more than a language; it is a meta language [3]. It defines the syntax rules for creating the markup languages for encoding instances of particular document or message types. One of the most important syntax rules for XML is that markups principally consist of a start tag and an end tag (in contrast to HTML). Furthermore, tag pairs can be nested within others to an unlimited degree. These two rules imply a tree-structure on so called well-formed XML documents. When designing a new XML language, the document structure – which tags are allowed and how tagged

elements are nested – is usually defined in a Document Type Definition (DTD). Figure 2 shows an example DTD and an XML file based on this DTD.

Since XML is a meta language, everyone can design a new language (or DTD) for his/her own purpose. Everyone is allowed to define his/her XML tags to delimit the data [12]. The interpretation of the data is completely left to the application that processes it, but is always based on tags and their names. Since XML files are text files, XML markups usually make sense to humans. Therefore, humans can read XML files. But they are not meant to be read by users. A user should never see the XML file. Human readable markups should give a hint to programmers who write code to process data included in XML files.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE LineItems [
<!ELEMENT LineItems (Item+)>
        <!ELEMENT Item (EAN, Description, Quantity)>
                <!ELEMENT EAN (#PCDATA)>
                <!ELEMENT Description (#PCDATA)>
                <!ELEMENT Quantity (#PCDATA)>
]>
<LineItems>
<Item>
                <EAN> 123 </EAN>
                <Description>Computer              XY</Description>
<Quantity> 3 </Quantity>
</Item>
<Item>
                <EAN> 456 </EAN>
                <Description>   Compter   ZX   </Description>
<Quantity> 5 </Quantity>
</Item>
</LineItems>
```

Figure 2: XML-Example

Accordingly, XML concentrates on the definition of how to represent the information being exchanged, while disregarding the programming details. The idea of separating document structure from document processing allows for XML's intent to "write once and publish everywhere" [3]. Consequently, an XML file of its own is worthless unless combined with a program that processes it. For example, latest versions of Web browsers can read XML documents, load an appropriate style sheet and format it for presentation on the screen. But the XML document can also be transformed into any other format, which might be more appropriate for printing or for presentations etc. Furthermore, there might be no need to display a document. The XML document can be sent over the Web and be handled by an accompanying Java applet or be processed by some software of the receiver – as it is in case of EDI.

It should be noted that XML itself is just the core of a standard family. The full set comprises among others XML Linking Language (XLink), XML Pointer Language (XPointer), XML Query, Extensible Stylesheet Language (XSL) and XML schema. The XML Schema standard will be important for EDI, because it will contain primitive and user-defined data types which makes it possible to express structural relationships between document types.

3. The Essence for Using XML for EDI

Since XML allows defining data interchange formats and it is possible to create DTDs that might be semantically equivalent to UN/EDIFACT message types, it is evident that XML is suited for the area of EDI.

One of the strong points of XML is its flexibility. Even non-technical experts are able to quickly develop a new (proprietary) DTD, which contrasts the long standardisation process in approving a globally accepted UN/EDIFACT message type. This might be the strongest argument for adopting XML based EDI solutions for closed communities where the meaning of the tags is clear to all partners. Furthermore, the DTD can be effectively distributed by publishing it on the Web.

But there are other topics where XML is superior to UN/EDIFACT. UN/EDIFACT was designed to transfer business-related data that can be presented in text format, but the transmission of non-textual information, like pictures or CAD data, is not possible today. With the concept of unparsed entities and notations XML offers an easy way to incorporate multimedia data that might support the business transaction. Additionally, XML opens the door to use the Internet as transmission medium rather than proprietary networks. Firstly, this will reduce significantly communication costs in EDI. Secondly, this allows the usage of the Unicode standard, a character encoding system supporting the intermingling of text in different languages [3].

Another benefit of XML results from its ability to serve as exchange format for different e-commerce solutions, in contrast to UN/EDIFACT's sole usage as exchange format between heterogeneous applications. Consider the following example: A bank offers an application-to-application interface for directing payments to its large customers, for medium companies it offers a home banking package, and small companies and end users can use electronic forms on the WWW. But UN/EDIFACT is only used as transfer syntax for the application-to-application interface, whereas the home package is based on a proprietary format and the WWW forms make use of a CGI interface. So the data format is different for the three transmission methods, but the transmitted semantics is the same in each case. By using XML as transfer syntax the three transmission methods can be streamlined: XML can be used in EDI environments, XML can be used in conjunction with WWW forms and XML can even serve as data format for proprietary applications.

So, XML offers a lot of benefits. But what about the benefit most often mentioned: "XML is easy to use even for SMEs, because XML is readable by humans"? The fact that XML can be read by humans includes another positive aspect. People feel more comfortable to look and play with XML, than it is the case for UN/EDIFACT. The popularity of the WWW will even strengthen this aspect. In a very short period there will be a lot of experienced programmers to write code for processing XML-based data. After 10 years the number of programmers experienced in building up an UN/EDIFACT based solution is still small.

But people shouting the above mentioned statement often mix up human readability with semantics. Machines are somehow stupid, they must be exactly told what the meaning of a tag is and accordingly how to process the data. But XML by itself governs syntax only. While XML markup makes it easier for humans to guess what the information between tags signify, the precise semantics cannot be specified in a computer-processable way.

Being able to invent whatever tags are needed in a DTD for a particular business document is not sufficient – one needs to know what the tags mean. Associating meaning with tags often requires additional mechanisms like a natural language

statement providing a precise definition. How should a machine know what the tag <*delivery date*> stands for? A programmer might have an idea what is meant by delivery date. But is it really sure that <*delivery date*> is used to describe the date of delivering products? Suppose yes – but delivery date is not semantically complete; there is still something left to the interpretation of the programmer. Is it a fixed or an expected delivery date? Is it an exact delivery date; is it the earliest date or the latest date for delivery? – Who knows?

```
<Store> Store A
        <Item>Item 1</Item>
        <Item>Item 2</Item>
</Store>
<Store> Store B
        <Item>Item 1</Item>
        <Item>Item 2</Item>
</Store>
```

or

```
<Item>Item 1
        <Store> Store A</Store>
        <Store> Store B</Store>
</Item>
<Item>Item 2
        <Store> Store A</Store>
        <Store> Store B</Store>
</Item>
```

Figure 3: N:M Relationships in XML

The structure of a DTD is obviously determined by the existing semantic constraints between the objects described. But, some decisions are still left to the DTD designer. A major design decision concerns the representation of n-to-m relationships between objects. Since DTDs force a tree structure on data, it is up to the designer to decide which objects will be nested in the definition of the other ones (see example in Figure 3). But the designer has to make further important decisions, as what will be presented in the data part of an XML file and what will be only an attribute to a tag. So it is easy to guess that two different DTDs for exactly the same business document covering exactly the same semantic will most likely differ in name and structure if created by two different designers.

Accordingly, if every company invents its own DTDs carrying its own tags, this will end up in a proliferation of proprietary DTDs for the same business transaction using different structures and tag names for the same semantics. Furthermore, it might be possible that in different DTDs the same name will be used for different semantics. But XML offers a namespace mechanism to avoid such naming conventions. Hereby an element will be associated with its authority [17]. Nevertheless, it must be mentioned that this namespace mechanism is optional, and therefore is not always used in DTDs.

The flexibility in creating new DTDs cannot be compensated by the easy access of the DTD via an URI on the Web. If a company has to consider different DTDs for the same business transaction, it will be no problem to view the documents in a browser (under the precondition that each DTD is associated with a style sheet). But when interfacing independent software a programmer has to write a specific code for each particular DTD. So the lack of a content standard obviously impedes interoperability.

To avoid the proliferation of proprietary XML languages, some organisations have started to co-operate in developing XML-based specifications for the business transactions in their particular sector or domain. Examples for those co-operations are RosettaNet, the Open Applications Group (OAG), Microsoft's BizTalk, Open Buying on the Internet (OBI) and the Open Travel Alliance. An acknowledged XML specification for a particular sector or domain is a step into the right direction.

Nevertheless, each industry solution ignores the other. Although some concepts, as delivery dates (from a data perspective) or invoicing (from a process perspective), are common to all domains, it seems they all reinvent the wheel. As a consequence, currently a number of sector or domain specific solutions do exist that are incompatible to each other. This situation reminds someone of the situation in EDI in the 70‘s and 80‘s. After realising the limitations of proprietary solutions, industry standards were developed, as ODETTE (European automotive industry) or TDCC (US transportation industry) to mention a few. But in EDI this was not the end of the story. The next step resulted in branch-independent standards like X12 and UN/EDIFACT.

4. The Future of XML for EDI

XML supports collaborative distributed Internet computing. Consequently, XML is suited for the application-to-application exchange of business data over the Internet. The interest in XML seems to be continuously growing and XML seems to be the first choice for defining data interchange formats in Internet applications. Although it is very unlikely that traditional EDI standards like UN/EDIFACT or X12 will disappear (too many well working EDI solutions already exist without a need to change), XML is a standard that organisations will have to consider when conducting business electronically. Traditional ‚EDI people‘ should beware of ignoring XML – may be they will be forced to use XML by their business partners.

But XML is not a solution by itself, because it covers syntax only. The fundamental problems of designing messages that meet business process requirements heavily depend on semantics and are independent of the syntax in which the messages are encoded [17]. The EDI community has a lot of experience in analysing the business needs of business transactions. Of course the EDI community made some mistakes, but their know-how should be considered as of extremely high value for XML and future developments. Even if the business knowledge is just written down in UN/EDIFACT or X12 syntax, which is hardly understood by an outsider, this knowledge should be transformed to provide a solid basis for XML.

But automatically generating XML tags and DTDs from UN/EDIFACT (and/or X12), would be not a proper solution either. Due to the hierarchical design of UN/EDIFACT a definition of a segment is independent of a message type, each code for a data element of the corresponding segment is valid in each message where the segment appears. May be the *slaughtering start date* is relevant for messages between a slaughterhouse and a ranger, but is it relevant when transmitting a patient’s record? It is obvious that auto-generation is not suited to capture the business knowledge of business transactions. Furthermore, transmitting the data values in the same order as in UN/EDIFACT but with tags instead of delimiters would only add to network cost without any benefit.

Today some companies claim that they have converted UN/EDIFACT into an XML presentation. That may be right, but this does not mean that they are compatible with each other. They are only proprietary solutions that try to migrate the UN/EDIFACT know-how. This is certainly a good concept, but also fails in true interoperability. Consequently, the best way for mapping UN/EDIFACT to XML has to be identified and - even more important – it must be accepted throughout the whole user base.

A good strategy would be employing the OO-edi [16] approach that utilises UML to achieve a top-down, process model driven approach to design business transactions. That means, that the knowledge captured by Message Implementation Guidelines

(MIGs) should be represented in transfer syntax-independent UML models. UML models are able to capture all the semantics included in business transactions [8]. These UML models should then be transformed according to a unique methodology into unambiguous XML DTDs by using a set of XML core components.

Repository and registry facilities will be key factors to put the above mentioned concept into work. An XML registry and repository facility addresses the challenges to interoperability as outlined in [5]: standard-based descriptive cataloguing, namespace management, name registration, resource authentication, robust name resolution, distributed resource maintenance, and education. The first global XML repository – XML.org Registry and Repository - is currently established according to the specifications of the Organization for the Advancement of Structured Information Standards (OASIS). This repository would certainly be a good candidate to keep the information included in the UML class diagrams in XML syntax. Nevertheless, it would be desirable to include also all the other UML notations in a repository, to describe the semantics as good as possible. The Object Management Group's (OMG) XML Metadata Interchange (XMI) standard [13] provides ways to represent the UML models in XML format. By utilising XMI the UML models can be stored in the repository and can upon be easily navigated by programmers.

Since a commonly agreed solution is desired in the EDI as well as in the XML community, the two major players UN/CEFACT and OASIS have joined forces to initiate a 18 month world-wide project, which started in November 1999. The electronic business XML (ebXML) project should lead to standardise XML business specifications. The success or failure of this project will certainly influence the future of XML in the area of EDI.

The purpose of ebXML – as defined in the terms of reference [18] – is to research and identify the technical basis upon which the global implementation of XML can be standardised. The goal is to provide an open technical framework to enable XML to be utilised in a consistent and uniform manner for the exchange of electronic business data in application-to-application, application-to-person and person-to-application environments. The scope of the ebXML initiative is to develop and publish, in the public domain, relevant and open technical specifications in support of domestic and international electronic business exchanges.

By endorsing the ebXML initiative UN/CEFACT put its own XML programme on hold. As 'owner' of the UN/EDIFACT standards UN/CEFACT recommends that a solution for migrating UN/EDIFACT to XML must be in-line with the outcome of the ebXML framework. Developing proprietary UN/EDIFACT-like DTDs must be stopped.

Conclusion

In a very short time XML has become the language for defining data interchange formats in Internet applications. XML is a meta-language suited to design self-defined markup languages. Accordingly, anyone can create his own XML based markup language by defining a (or a set of) DTD(s). The newly created business markup language can be used as message type for exchanging business documents among business partners. This approach provides some advantages compared to the traditional EDI standards approach. DTDs can be immediately created without a long standardisation process. The whole world can access the DTD by putting it on the WWW and assigning a URI. The usage of the Internet as transport medium provides a

low cost solution and the usage of the Unicode standard. The popularity of the Internet will generate a lot of experienced XML programmers in a very short period. Due to the decoupling of data presentation and processing, XML files can be used in many different ways: they can be viewed in a browser, processed by an accompanying Java applet or by independent software.

But there is a very important precondition that must be met. Everyone participating in the exchange of XML files must be aware of the meaning of tags. This works well when business documents are exchanged within a closed community. When organisations must communicate with others living in ‚other XML-worlds' the limitations will soon be recognised. Machines are not able to understand the meaning of self-defined tags and programmers must implement a specific code for a particular message. Therefore, semantic transparency is needed. This means that machines and humans are presented with information that is both unambiguous and meaningfully correct [5].

The EDI community has a long experience in capturing the business needs of business transactions. This experience should be used to create syntax independent design rules that can be used as basis for a unique transformation into an XML presentation for repository storage. The definition of a single global market on the basis of XML is the goal of the ebXML initiative of UN/CEFACT and OASIS.

EDI is not different to many other applications in the world: flexibility and standardisation stay in opposition. Flexibility allows short-time solutions whereas standardisation ensures interoperability.

References

1. Berge J. (1991). The EDIFACT Standards. NCC Blackwell Limited, Oxford
2. Bosak J. (1997). XML, Java and the Future of the Web. World Wide Web Journal (W3J), O'Reilly
3. Bosak J., Bray T. (1999). XML and the Second-Generation Web. Scientific American, May 1999. http://www.sciam.com/1999/0599issue/0599bosak.html
4. Cover R. (1998). The Essence and Quintessence of XML. Retrospects and Prospects. OASIS, http://www.oasis-open.org/html/essence_of_xml.html
5. Cover R (1998). Managing Names and Ontologies: An XML Registry and Repository. OASIS, http://www.oasis-open.org/html/registry_and_repository.html
6. Emmelhainz M.A. (1990). Electronic Data Interchange: A Total Management Guide. Van Nostrand Reinhold, New York
7. Hill N.C., Ferguson D.M. (1989). Electronic Data Interchange: A Definition and Perspective. EDI Forum: The Journal of Electronic Data Interchange, Vol. 1, Issue 1, pp 5 – 12
8. Huemer C. (1999). Modeling Inter-Organizational Systems with UML. Proceeding of the 12th International Electronic Data Interchange Conference. Bled, Slovenia
9. Knoppers J.V.Th. (1992). Importance of the Open-edi Reference Model from a User and Business Perspective. Proceedings of the 5th International Electronic Data Interchange Conference. Bled, Slovenia

10. ISO (1996). The Open-edi Reference Model. ISO/IEC JTC1/SC30 IS 14662

11. Leventhal L., Lewis D., Fuchs M.(1998). Designing Xml Internet Applications. Charles F. Goldfarb Series, Prentice Hall

12. Megginson D. (1998). Structuring Xml Documents.Charles F. Goldfarb Series, Prentice Hall

13. OMG (1999). An Overview to the XMI - XML Metadata Interchange Specification. http://www.omg.org/news/pr99/xmi_overview.html

14. Peat B., Webber D. (1997). Introducing XML/EDI – „the E-business framework". XML/EDI Group. http://www.geocities.com/WallStreet/Floor/5815/start.htm

15. Raman D. (1996). Cyber Assisted Business – EDI as the Backbone of Electronic Commerce. EDI-TIE B.B., Hoofddorp, Netherlands

16. TMWG (1998). Reference Guide – "The Next Generation of UN/EDIFACT" (Revision 12). CEFACT/TMWG/N010/R12

17. TMWG (1999). TMWG Recommendations on XML. CEFACT/TMWG/N089/R5

18. UN/CEFACT, OASIS (1999). ebXML Terms of Reference. http://www.ebXML.org