



XML based supply chain integration at the Los Angeles and Long Beach ports

Final Report

METRANS Project 08-11

January 2010

Principal Investigator: Burkhard Englert

Professor

Co-Principal Investigator: Shui Lam

Professor

Carroll Chiou

Graduate Student

California State University Long Beach

Department of Computer Engineering and Computer Science

1250 Bellflower Boulevard

Long Beach, CA 90840

Tel: (562) 985-7987

Fax: (562) 985-7823

Email: benglert@csulb.edu

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, and California Department of Transportation in the interest of information exchange. The U.S. Government and California Department of Transportation assume no liability for the contents or use thereof. The contents do not necessarily reflect the official views or policies of the State of California or the Department of Transportation. This report does not constitute a standard, specification, or regulation.

Abstract

In this project we perform a cost / benefits analysis of EDI and XML based communication between shipping lines, terminal operators, government agencies, trucking companies, rail operators and other agents at the Los Angeles and Long Beach ports. We analyze the suitability of (1) XML/EDI, (2) EDIINT (Web EDI), (3) Collaboration EDI, (4) The language M and (5) Web Services for use as communication platforms at the ports.

We conclude that currently at the Los Angeles and Long Beach ports information is mostly shared in a bilateral manner. As a result we describe and recommend an approach where terminals share data with shipping lines / carriers, rail operators, trucking companies, truckers and government agencies using XML based Web Services. This approach allows agents to either make information accessible through a web browser or through direct computer system to computer system communication. In the latter approach the terminals computer system will function as an automated information clearing house that can provide up-to-date, real-time information to all interested and authorized parties. Web Services furthermore allow securing and authenticating this information, protecting its dissemination. We believe that this approach can lead to increased throughput at the ports and an increase in capacity.

Table of Contents

1. Introduction.....	1
2. Background and Motivation	2
3. Electronic Data Interchange (EDI).....	3
3.1 Why EDI?.....	4
3.2 ASC X12 and EDIFACT	5
4. The Extensible Markup Language (XML)	7
5. XML/EDI.....	8
5.1 Internet EDI.....	10
5.2 Basic XML/EDI Systems.....	10
5.2.1 Types of XML/EDI applications	12
5.3 Collaboration XML/EDI	16
6. XML conversion	17
6.1 XML conversion methods.....	18
6.1.1 BOTS open source EDI Translator	19
6.1.2 m-e-c eagle	19
6.1.3 Microsoft Biztalk 2006	20
6.2 Conversion methods for XML style sheets.....	20
6.2.1 XSL Conversion	21
6.3 XML-EDI Conversion Experiments.....	22
7. EDIINT – EDI over the Internet	23
7.1 EDIINT Security	24
7.1.1 bTrade TDAccess/TDPeer	25
7.1.2 Inovis BizConnect.....	25
7.1.3 Gentran Integration Suite	25
7.2. Problems with EDIINT (and Web-EDI)	26
8. The Language M	27
8.1 Comparing M and XML	28
9. Communication Model at the Los Angeles and Long Beach Ports	29
9.1 Terminal Operating Systems – Navis Sparcs N4	29
9.1.1 Groovy.....	31
9.1.2 Announced features of Navis Sparcs N4.....	32
10. Navis Sparcs and Web Services	33
10.1 Integration through Web Services	35
10.1.1 Integration with trucking companies.....	35
10.1.2 Integration with rail facilities	36
10.1.3 Integration with container depots	36
10.1.4 Integration with shipping lines	36
10.1.5 Integration with Port Communication Systems	37
10.1.6 Internal Integration.....	38
10.2 Implementation of a Web Service connection	38
10.2.1 Consuming a Web Service	39
10.3 Web Service Security.....	40
11. Recommendations and Conclusions	41
12. Implementation.....	44
13. References	46

List of Figures

Figure 1: An excerpt from an ASC X12 purchase order document	6
Figure 2: An excerpt from an EDIFACT orders document	6
Figure 3: An excerpt from a RosettaNet PIP 3A4 Purchase Order request document.....	7
Figure 4: Information flow in traditional EDI systems	8
Figure 5: Biz talk Mapper Example	10
Figure 6: Direct connection of two EDI Servers.....	13
Figure 7: Direct connection scheme between client and EDI server	14
Figure 10: XSL conversion for screen display	21
Figure 11: XSL conversion for printing	21
Figure 13: Communication structure using M.....	28
Figure 14: Current use of EDI at the ports of LA and LB	29
Figure 15: Navis Screen shot showing vessel profile [42]	30
Figure 16: Navis Screen shot showing Container information [42]	31
Figure 17: Web Services as portals to PCS	37
Figure 18: Web Services enabled access to a Terminal Operating System (Phase 1)	45
Figure 19: Web Services enabled computer to computer communication (Phase 2)	46

List of Tables

Table 1: Comparison of EDI and XML.....	2
Table 2: Comparison of EDI transmission formats.....	11
Table 3: Comparison of costs of EDI transmissions	12
Table 4: Types of XML conversion	18
Table 5: Cost/benefits of EDI / Web Services	44

Disclosure

Project was funded in entirety under this contract to California Department of Transportation.

1. Introduction

In recent years internet related technologies such as XML have created new opportunities for electronic communication between different companies. At the same time many companies have changed their business operations in ways that could benefit greatly from increased communication opportunities. However, for historical and other reasons the use of this technology for communications has been limited at the Los Angeles and Long Beach Ports. Like many other companies terminal operators at the Los Angeles and Long Beach ports generally use EDI (Electronic Data Interchange) to communicate electronically with other companies. While EDI is compact, it is difficult to maintain and extend and generally only allows computer to computer communication. Moreover it requires large financial investments in proprietary systems and software. As a result its use is often confined to larger companies. At the ports, for example, EDI communications usually does not include smaller customers of terminal operators or truck drivers limiting its usefulness. For this reason EDI alone cannot be the basis of port communications. Platforms and systems are needed that allow all agents at the ports – shipping lines/carriers, terminals, rail operators, government agencies, trucking companies and truckers to effectively participate in this communication and exchange.

In many other areas where effective business to business communication is essential to support operations XML has become the underlying language that can support this communication.

XML based communications are accessible through any web browser and are thought to be cheaper than EDI based communications. A perceived lack of interoperability however remains a drawback of an electronic communication system that is solely based on XML. While it is not realistic to expect companies to switch from EDI to an exclusively XML based communication platform in the immediate future there are tools available or can be designed that allow to translate between EDI and XML and between XML and EDI efficiently. As a result there are several XML based options one could pursue: (1) XML/EDI, (2) EDIINT (Web EDI), (3) Collaboration EDI, (4) The language M and (5) the use of Web Services.

In this project we perform a closer investigation of all these options. We determine in which way XML can be best used so that it provides the best possible, communication support that is needed for effective and efficient communication and exchange of information at the ports. We will compare the cost and benefits of these options.

The use of EDI is very widespread among shipping lines and terminal operators. One example of its use is the stowing plan of a container ship before the ship's arrival at the port. Small enterprises that operate at or near the port on the other hand generally do not use EDI due to high financial cost requirements and implementation difficulties.

Companies that have invested into EDI would be reluctant to redesign their business communication technologies just because a new technology appears. It is important to note that there is no reason why the investment a company made into EDI cannot be embedded into an XML based communication. All the semantics, data sets, business vocabularies, code lists and processes could be mapped to an XML messaging standard as illustrated in the table below.

Feature	EDI	XML
Origin	trade	internet based
multi-channel support	only computer to computer	Yes. Also human readable
modus of communication	1:1	1:1 /n : m / combinations
Focus on	compactness	interoperability
Tool support	Little; mostly proprietary	high and open source
Standards	UN/EDIFACT; ANSI-X12	ebML, Bolero XML

Table 1: Comparison of EDI and XML

2. Background and Motivation

In 2007 the total number of containers handled at the ports of Los Angeles / Long Beach actually declined by 0.2% [32]. In 2008 the downturn accelerated with a decline of 8.5%. For 2009 a further decline by 13.3% has been forecast. For 2010, however a growth in exports and imports is expected [32]. Despite the current downturn and decline to remain viable as the center of International trade for the United States the L.A. / L.B. ports and the Southern California region must look for alternative growth opportunities. Consequently existing facilities and infrastructure will have to handle a significantly increased load. The lack of space, shortage of money and an increased awareness of the impact of the port and its supply lines on air pollution render it difficult and potentially impossible to absorb this projected growth simply through a parallel growth in infrastructure. New approaches that are cost effective, that increase the efficiency of operations and that at the same time limit or possibly even lower the amount of congestion and air pollution are needed. To be realistic and ultimately effective these approaches must start with the current operating conditions at the ports and a systematic plan to further optimize them. As it was shown at other large ports (e.g. Rotterdam)[58] one such promising area of future optimizations are the electronic communications between terminal operators and customers, shipping lines, truck drivers and US customs. Currently port terminals in general use EDI (Electronic Data Interchange) to communicate with carriers (for example to exchange a stow plan in advance) and their largest customers. EDI is a compact protocol that allows for one on one computer to computer communication but because it is built on proprietary technology comes with a large price tag. This means in practice that most small companies including those owner operated trucking companies cannot participate in this form of communication. Moreover, the EDI standards are more like a library or a tool kit than the definitions of a fixed protocol. This means that for every new communication that is set up a substantial amount of testing and fine tuning is required resulting in substantial overhead costs, again limiting the scope of EDI communications at the ports.

On the other hand experiences at other ports [58] show that enabling all parties to participate in port communications has a very positive impact on the efficiency of the port and the connected supply chains as a whole. At the Los Angeles and Long Beach ports there are some initial steps in this direction. Some terminals for example, have instituted appointment systems that allow truck drivers to set up a one hour pickup window, reducing wait and idle times at the port and consequently congestion and

ultimately pollution. Newer Terminal Operating Systems (TOS) enable terminals to potentially give shipping lines / carriers, rail companies and truckers/trucking companies partial access to their computer systems.

In this project we study in which way and to what extent this initial effort could be best extended. We study in which way XML based communication could be best used to enhance connectivity at the ports. Considering the sizeable past investments into EDI we do not expect large companies to completely move away from EDI any time soon. We however investigate how and at what cost these EDI communications could be embedded into a larger communication system that is based on XML. To do so we investigate the possibilities of translations between EDI and XML and XML and EDI, EDIINT, the language M and look into turn key solutions such as XML (Web Services) enabled terminal operating systems.

Many people believe that XML could be used to accomplish the same things as EDI with much cheaper technologies and infrastructures. If this is correct it will allow many more companies that operate at the ports – such as small trucking companies, smaller terminal customers – to invest in such technologies. In the end this will lead to greatly enhanced port communications leading to a significant reduction in congestion, pollution and hence cost and to a marked improvement in efficiency. We believe that it is likely that both EDI and XML communication will coexist at the ports through transformations between them. Therefore it will be all the more important to use the right transformation tools and mapping definitions.

The installation of Web-based appointment systems at the Evergreen and other terminals at the ports is clear evidence of the anticipated benefits of an increased electronic communication at the ports. Moreover since XML also – though albeit smaller – has certain interoperability issues and requires the translation between different XML forms – we extend our study to include other XML based protocols and platforms such as the language M and Web Services. Finally we consider the security of the approaches that we discuss. It will be crucial to ensure that only authorized users can access information that is disseminated by legitimate partners. We will now first explain the relevant communication technologies, EDI in chapter 3, XML in chapter 4, EDI/XML in chapter 5, XML conversion in chapter 6, EDIINT in chapter 7, the language M in chapter 8. We then describe the communication model at the LA and Long Beach ports in chapter 9 and Terminal Operating Systems and Web Services in chapter 10. We finally conclude with some discussion and recommendations in chapter 11.

3. Electronic Data Interchange (EDI)

EDI is a standard format for exchanging business data. EDI's goal is it to implement the automated, electronic exchange of standardized, structured and normalized messages from computer to computer between different organizations in commercial or administrative transactions. There are many different standard EDI messages that have been developed. To enable business communications in many different situations these standard messages are very flexible and are usually interpreted using some sort of tool box. This means that every time a new communication is set up the standards must be adjusted to the given situation requiring careful negotiations between the involved parties and thorough testing. In general this makes EDI a labor intensive technology. Depending on the background, experience and requirements this can take several hundred human

hours on both sides. Besides not being human readable the EDI input to application programs also requires some translation or conversion.

In the shipping and transportation industries two standards are prevalent: the American Standards Institute's X12 (ANSI-X12) standard [34] and the United Nations Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) standard [3]. EDIFACT is the predominant international standard while X12 is used mainly in the United States. Container terminals and shipping lines have been communicating via EDI for many years. The APM and ITS terminals at the Long Beach port for example, uses X12 to communicate with carriers.

The use of different EDI standards is also very common. Hence if two companies want to start up EDI communications they first must negotiate an agreement to define the subset of EDI they want to use [26]. This agreement is called an implementation guideline. As a result, EDI messaging standards are frequently modified and hence diverge significantly [44, 45]. This leads to an equally wide range of available software and systems to support their use. Most commercial users buy a package which typically handles (1) mapping between the EDI standard and internal file formats defined by the user; (2) the management of trading partner relationships by maintaining a database of who's who and what messages are enabled as part of the trading agreement; (3) the timetabling and automatic running of online sessions to send and receive messages [7, 11, 38].

The Internet is increasingly used to send and receive EDI messages, either by FTP [28] or by email transfer. In 1995 a simple MIME protocol for encapsulating an EDI message was defined [15] and, under the leadership of the Internet Engineering Task Force, a much more comprehensive standard is being finalized with the goal to enable an EDI message to be sent over the Internet with the same levels of security and audit trail which Value Added Networks have traditionally provided.

3.1 Why EDI?

Given these disadvantages it may seem as if there are only very few reasons to use EDI at all. This however does not explain why EDI is still so widespread especially among large companies in the transportation industry. Before EDI mainly manual keyboard data entry was used to process incoming shipping documentation. Because of a lack of standard documents that were flowing through the transportation cycle and across the globe human actions were the only possibility for processing this information. Data entry operators would quickly scan a document and find the information they would want to capture. The operator would only capture the information needed at this point of the shipping cycle all other information potentially needed at later stages of the cycle was lost. Companies could potentially outsource this operation but this does not remove its inherent inefficiencies. A medium or large carrier or shipper would have to employ 100 or more data entry operators to process all documents in this manner. Because data is copied and entered manually errors were very common. Studies showed that employees would spend about 10% of their time correcting errors and that a skilled operator made a mistake on average every 300 characters [48]. Errors are also the result of incorrect or illegible forms or of transmission errors.

EDI was adopted by the transportation industry in the late 1960's to allow for seamless, namely efficient and less error prone transmission of data between carriers and shippers. Data could now be transmitted directly from the shippers to the carriers and terminals

computer system. Standardized EDI documents such as **EDI-856 Advance Ship Notice/Manifest** eliminate the need for paper documents. Based on the Accredited Standards Committee (ASC) X12 format, the **856** is used to list the contents of a shipment of goods as well as additional information related to the shipment, such as order information, product description, physical characteristics, type of packaging, marking, carrier information and configuration of goods within the transportation equipment. The transaction set provides an ordered flexibility to convey information, enabling the sender to describe the contents and configuration of a shipment in varying levels of detail. The sender of this transaction is the organization responsible for detailing and communicating the contents of a shipment, or shipments, to one or more receivers of the transaction set. The receiver of this transaction set can be any organization having an interest in the contents of a shipment or information about the contents of a shipment.

So why then is EDI currently only used by an estimated 2% of the world's businesses? In fact it is mainly used only by large companies. Small and medium enterprises mostly stayed away from EDI for several reasons: First the set-up process for EDI is very complex. To send and receive an EDI file internal data formats such as database formats must be mapped and converted precisely to the required EDI standard. This process can either be taken care of in house by a dedicated EDI IT department or can be handled by an outside contractor. In both cases significant additional costs arise for the company. Also if the company chooses to select an outside contractor to handle its EDI traffic it usually gets locked into a vendor's proprietary system and Value Added Network (VAN). Hence for medium size and small companies EDI was never really an option. This means that while EDI is an efficient and effective means of electronic communication it always only represented a small part of the global business to business communications. To make EDI a viable option for comprehensive global supply chain communication the mapping of EDI documents to other data formats must be addressed. To do this a company could either internally develop its own parser or buy a parser from one of the many vendors.

3.2 ASC X12 and EDIFACT

The Transportation Data Coordinating Committee (TDCC) was formed in 1968 to standardize the electronic exchange of data for all transportation industries in the US. The American National Standards Institute (ANSI) continued this effort in 1979 and began to develop ASC X12 (Accredited Standards Committee) whose first version was released in 1982. EDIFACT on the other hand is based on the recommendations of the United Nations Economic Commission for Europe (UNECE). ISO approved EDIFACT as an international standard in 1987. In 1992 ANSI announced that the continued development of ASC X12 would end in 1997. Many US companies subsequently, however did not see any benefits in switching to EDIFACT. As a result X12 and EDIFACT are now both continuing to evolve as two independent standards. This will likely remain the case for the foreseeable future.

We will illustrate the two EDI formats with the very simple example of a purchase order. According to this order a delivery will be shipped to "EDICustomer" in the organization "CSULB" at the street address of "1250 Bellflower Blvd.", with the postal code "90840" in "Long Beach" in the "USA". Figure 1 shows the representation of this information in

the ASC X12 format, while Figure 2 shows the representation in the EDIFACT format. ASC X12 specifies the segment codes “N1”, “N2”, etc, EDIFACT the segment code “NAD” and both use the element values “ST” and “US”.

```
N1*ST*CSULB~  
N2*EDICustomer~  
N3*1250 Bellflower Blvd.~  
N4*Long Beach**9840*US~
```

Figure 1: An excerpt from an ASC X12 purchase order document

```
NAD+ST++CSULB+EDICustomer+1250 Bellflower Blvd.+Long  
Beach++90840+US'
```

Figure 2: An excerpt from an EDIFACT orders document

An EDI document does not contain any information about its structure. Therefore sender and receiver must agree in advance on which standard and which standardized document type they are using. Also a company may decide not to use a field provided by the standardized document format. The company must then ensure that its communication partners are aware of this. Otherwise any information sent in this field will be lost or misunderstood. Finally business software can generally not understand native EDI documents therefore some sort of mapping, i.e. conversion between native data formats and EDI must occur. For example to populate an EDI document certain database entries must be entered into the correct fields of the document. This explains the need for continued human supervision and intervention for any EDI system. In addition EDI standards are based on business needs. Since business needs tend to change rapidly EDI standards to be effective and supported by the business community must be able to change together with the business processes. As a result EDI standards can change up to two times a year. An investment into EDI is hence never a one time investment. An investment into EDI is also an investment into the information system that supports its use. Companies that use EDI for business transactions must either support an internal EDI department that monitors and implements these changes or hire a contractor that provides all EDI communication and conversion via a value added network.

This means that EDI is tied to large investments and maintenance costs explaining why mostly only large companies use it [2, 18]. On the other hand EDI is very effective for business transactions: It allows for direct business to business communication without human intervention. The EDI file sizes are usually very compact. Since EDI files do not contain information about their structure they are usually relatively small – most files are less than 50 Kbytes. Finally since companies have already made large investments into this technology they are looking for a return on their investment. This means that any technology wanting to replace EDI must come with verifiable benefits that impact the bottom line of a company directly.

The literature also shows that EDI based business interactions provide many advantages over manual business interactions. EDI provides a speedup of business interactions [20] it reduces the number of errors [54] and operating costs [41]. But companies often use EDI with a small fraction of its business partners and for only a small subset of transactions with these business partners [49]. A terminal operator, for example, in most cases uses

EDI only for the transfer of manifests between carrier and terminal, rail operator and terminal and customs and terminal. EDI is used for interchange in both directions. Small and medium size companies often lack the organizational readiness, are afraid of the EDI costs, their customers and a majority of their partners do not use EDI or the volume of business transactions is too small to justify the investment into EDI [54].

4. The Extensible Markup Language (XML)

XML [64] is designed to improve the functionality of the Internet. It is a text based syntax standard that acts as a foundation for the development of semantic standards. In XML, a user can specify the structure of a document in an extensible Document Type Definition (DTD), a file that determines how mark up tags should be interpreted by an application that presents the document. XML strictly separates structure and content, so a DTD can be used for several XML documents as a template. The presentation itself is specified in a style sheet. This allows a user to present the same content in many different ways (using different style sheets) without having to reorganize the content. XML is supported by many programming languages, applications and inexpensive tools such as web browsers making it a technology that is easy to access and use.

```
<shipTo>
  <BusinessDescription>
    <businessName>
      <FreeFormText xml:lang="US">CSULB</FreeFormText>
    </businessName>
    <PartnerBusinessIdentification>
      <ProprietaryDomainIdentifier>EDICustomer</ProprietaryDomainIdentifier>
      <ProprietaryIdentifierAuthority>CSULB</ProprietaryIdentifierAuthority>
    </PartnerBusinessIdentification>
  </BusinessDescription>
  <GlobalPartnerClassificationCode>End User</GlobalPartnerClassificationCode>
  <PhysicalLocation>
    <PhysicalAddress>
      <addressLine1>
        <FreeFormText xml:lang="US">1250 Bellflower Blvd.</FreeFormText>
      </addressLine1>
      <cityName>
        <FreeFormText xml:lang="US">Long Beach</FreeFormText>
      </cityName>
      <GlobalCountryCode>US</GlobalCountryCode>
      <NationalPostalCode>90840</NationalPostalCode>
    </PhysicalAddress>
  </PhysicalLocation>
</shipTo>
```

Figure 3: An excerpt from a RosettaNet PIP 3A4 Purchase Order request document

The example in Figure 3 [44] shows the same purchase order form Figure 1 and Figure 2 in XML format. The Rosetta Net [46] Partner Interface Process (PIP) defines the element names “shipTo”, etc. and the contents “End User”, “US”, etc., the attribute name “xml:lang” and the attribute value “US”.

Many messaging standards are based on XML. In the messaging context the content of a message is represented in an XML document that conforms to the definition in the XML scheme. An XML scheme describes the structure and data typing of one specific class of XML documents, its default values and documentation. The scheme can also be used to validate XML messages, that is ensure that a given XML documents conforms to the rules specified in the scheme. This naturally allows the use of many different standards and as a result the transformation between these standards must be addressed. This is one

of the weaknesses of XML. One possible solution is the use of the **XSLT** transformation language. This language allows users to adjust content towards humans or devices with different preferences or capabilities and to devices with different semantics [52].

There are also tools available that allow the transformation of EDI messages into XML and vice versa. In an EDI/XML world a customs agent, for example, could update the status of a container in a central database through a mobile device that uses a Wireless Markup Language (XML for wireless devices). An authorized truck driver could then – using a web browser on his PDA (through XML) - request an update on the container from the central database. At the same time a data export module that is attached to the central database could automatically notify on the basis of EDI or XML/SOAP[9] a terminal operator of any changes to the status of the container.

There are many different XML messaging standards for e-business. In electronic commerce we have cXML[13], BMEcat[6], ebXML[19], in supply chains RosettaNet[46] or BizTalk[5], in the transport sector Bolero[7] and tranXML[56].

Several studies show that XML has clear advantages compared with EDI [26, 29,47,50]. On the other hand many researchers believe that XML will not replace EDI in the near future [35, 60]. The main reason given is that the costs of XML dominate its benefits. In 2003 9% of companies in Europe that use computers used EDI-based e-business frameworks. In 2005 that figure went up to 19%. On the other hand in 2003 8% of these companies used XML based e-commerce frameworks, while 14% used XML in 2005 [16, 17].

Why does EDI - even though XML seems better suited to meet today's communication demands - appear to have an advantage over XML in terms of business adoption? A study by Farrell and Saloner [21] argues that standardization as in the case of EDI is *efficient* when its users are certain about the other user's benefits and costs even though they have different preferences. When users are unsure about others benefits and costs a "deadlocked" state results where they may become locked into inferior technology. The technology that initially is superior – EDI in our case – has an advantage even though it is not the optimal technology. To become the main technology XML based e-business frameworks will have to overcome this difficulty.

5. XML/EDI

In traditional EDI systems EDI documents are exchanged over a Virtual Private Network (VPN), through FTP or using AS2 (an Http based protocol) between two Enterprise Resource Planning (ERP) servers.

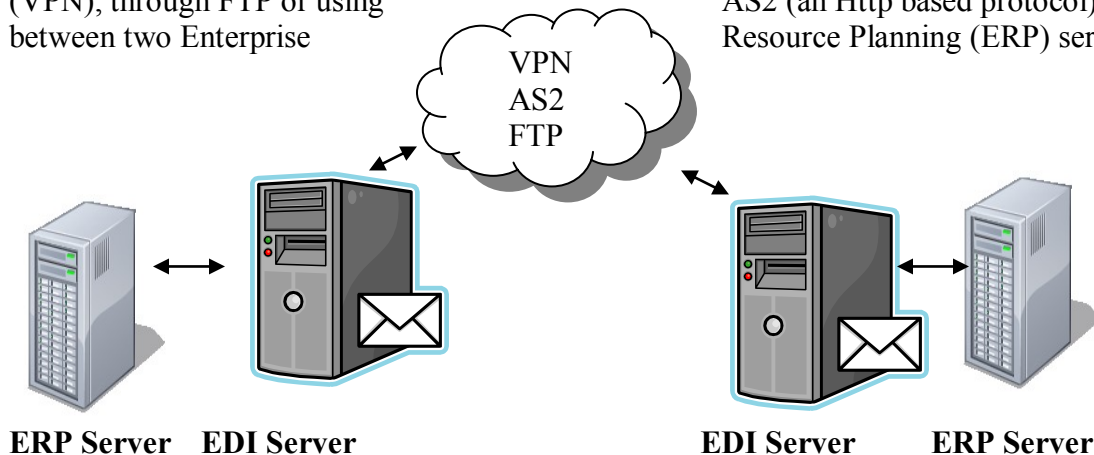


Figure 4: Information flow in traditional EDI systems

Over the last two decades the Internet has become a global and cheap electronic communication medium. Naturally this leads to the question whether one can use this cheap and ubiquitous network to transfer EDI files. The ISIS European XML/EDI pilot project was sponsored by CEN/ISSS to study the feasibility of XML for electronic data interchange, and was completed in January 2000 [33].

In scope, it examined conversion from UML- and EDIFACT-based systems for healthcare and transportation, respectively. It also studied the utility of auxiliary XML processes and specifications (such as XSLT) to determine what components may be missing from XML tools today. The lessons learned from both the UML- and EDIFACT-based investigations included:

- XML is capable of electronic data interchange using currently available tools.
- Original standards need to be simplified when converted to XML, such as normalizing data, removing codes, defining defaults and subsetting.
- General structures need to be converted to hierarchical structures, often with rules to facilitate automatic implementation.
- Mnemonics and programming-style names need to be edited to produce meaningful, human readable tag names.
- Chains of XSL Transformations allows application tailoring and simplifies applications by supporting localized XML DTDs, converting between forms (EDIFACT, WML, local format, etc.) and presenting as HTML.
- While the current set of specifications are adequate (XSLT, DOM, XML Path, and XML Schema), several necessary improvements were proposed.

XML's biggest weakness is its lack of standardization. To use XML together with EDI will only be possible if standardization succeeds. Efforts in this area include RosettaNet[46] and ebXML[19], which we will discuss in detail later (chapter 8).

While in EDI transformation is expensive, proprietary and cumbersome, tools and middleware in XML are generally cheaper and very powerful. Commercial systems such as BizTalk mapper[4] can be used to transform between EDI and XML. Figure 2 shows the configuration of such a mapper. In this example the source file is an EDI-based document, and the destination file is a flat-file document. Here the EDI document structure is first converted to an intermediate XML format, the structure of which is represented by an XDR (XML Data-reduced) specification. A data-driven parser creates an XML version of the EDI specification. The XSL engine then transforms this source XML representation to an XML representation of the destination file format. The destination specification is later serialized to the native format of the destination file, a flat file in this example. Creating mapping definitions, however, is still time consuming work.

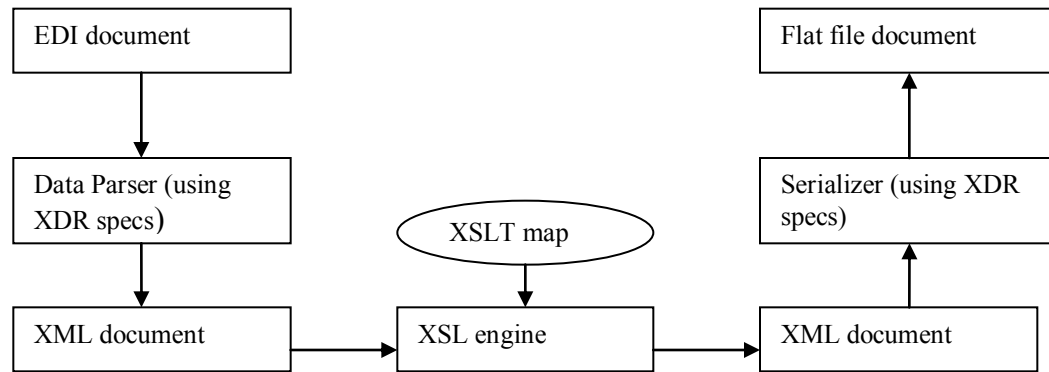


Figure 5: Biz talk Mapper Example

Value added Network based EDI has a long history. Since 1996, however, companies have begun to use the Internet to transmit EDI messages. There are three levels of this Internet based EDI:

- **Internet EDI** -- Transmission of EDI messages over HTTP, FTP, SMTP, or specialized banking industry protocol
- **Basic XML/EDI** – Representation of EDI messages in XML, and transmitted over Internet protocols (HTTP, FTP, SMTP)
- **Collaboration XML/EDI** – Exchange of XML messages defined by business processes, with messaging formats based on larger framework specifications (RosettaNet[46], ebXML[19])

5.1 Internet EDI

In recent years the area of **Internet based EDI** has emerged since the implementation and operating cost are relatively cheap. There are several problems associated with Internet EDI. The first problem is interoperability. Second it is difficult to connect automatically to the backend systems of the users, namely these backend systems can often not parse EDI files. Also if EDI files are sent over the public Internet, files potentially may become compromised. It would hence be desirable to add some error correction and authentication features such as digital signatures to the files or encrypt the files all together. XML based Web-EDI or XML/EDI addresses these weaknesses. In XML/EDI EDI files are described using XML syntax. The resulting file is then said to be in XML/EDI format.

5.2 Basic XML/EDI Systems

I. In basic systems files are either transmitted over FTP, TCP/IP or http/https.

1. FTP (File Transfer Protocol)

The most basic file transfer protocol in TCP/IP networks.

- Protocol is widely considered to be insecure so messages (files) are often unable to pass through firewalls.
- Not commonly used as an Internet EDI protocol.

2.TCP/IP

Allows use of a low cost and fully duplex modem.

- Messages can pass through firewalls.
- More reliable than FTP – error checking, redundancy, acknowledgments.
- Commonly used in Internet EDI.

3.HTTP/HTTPS

HTTP (Hyper Text transfer Protocol) – Developed for the exchange of html documents between servers and clients over the WWW.

HTTPS – HTTP equipped with an encryption function that is based on SSL (Secure Socket Layer) technology.

- Files can generally pass through firewalls.
- Low security risks (https).
- Becoming the mainstream communication protocol in Internet EDI.

II. Files can also be transmitted using email as attachments in this case the protocols used are SMTP or MIME. SMTP is the most basic mail transfer protocol. MIME also allows transmitting and receiving binary data.

III. Finally it is possible to convert a business document into an HTML document and post it as a web document to which the receiver then has access. This receiving process can also be automated.

	Advantages	Disadvantages
File Transfer	Can automatically connect to corporate system.	<ul style="list-style-type: none">• Specialized skills necessary to build and maintain system.• Expensive.
Email	<ul style="list-style-type: none">• Can automatically connect to corporate system.• No server necessary.	<ul style="list-style-type: none">• Need email based EDI software package.• Email is unreliable.
Web	Easy to operate	<ul style="list-style-type: none">• Requires human operation on the client side and data entry into the backend system.• Interface depends on each server. Client potentially needs to perform several conversions.

Table 2: Comparison of EDI transmission formats

The costs of the different formats also must be taken into account:

	General Costs	Application
File Transfer	Internal System Setup including EDI server estimated at up to \$10,000. Internet EDI ASP services: \$100 - \$300 + Internal EDI translator \$5000.	<ul style="list-style-type: none"> • Chosen when automatic connection is needed. • Good for large number of transactions (100 per day or more).
Email	Email based EDI software product - \$1000	Can be used with 100 or more Transactions per day in case of Automatic connection otherwise With several per day.
Web	Client's expenses depend on server's policy. With ASP service cost about \$100 per month.	Feasible with several transactions per day

Table 3: Comparison of costs of EDI transmissions

5.2.1 Types of XML/EDI applications

Based on the level at which XML is employed we can distinguish the following XML/EDI types:

1. Simplified basic XML/EDI

Selected business documents are partially converted to XML as a supplementary function to Web-EDI. Documents that are used with low frequency (e.g. monthly reports) are downloaded as XML documents. This requires the use of XML translation. Documents that are used frequently are transmitted via regular Web-EDI. This facilitates the introduction of XML/EDI and reduces performance problems.

2. Full scale basic XML/EDI with all business documents converted into XML.

Common and frequently used business documents are also converted to XML (e.g. purchase orders, receiving documents). In addition Web-EDI's file transfer, email or web transmissions are available. System can be built in house or by employing an ASP service. An XML style sheet can be used to customize screen displays and printing sheets. Extensibility of XML allows a company to prepare for future developments. Solves the conversion problems encountered in pure Web-EDI.

3. Full scale basic XML/EDI with a native XML database.

Extends a full scale basic XML/EDI system with a backend database that is also in XML format. This allows for seamless connection with other backend systems in the company (e.g. accounting system). The database however must be designed with security and performance assurance to coordinate with other functions.

We will now look at three distinct XML/EDI architectures. First we consider a system where servers are connected as peers and exchange XML/EDI messages (Figure 6). We then consider an architecture where a client is connected to an EDI server via XML/EDI (Figure 7). Finally we consider the case where a client connects to an EDI server via XML/EDI and an ASP (Application Service Provider) (Figure 8) [65].

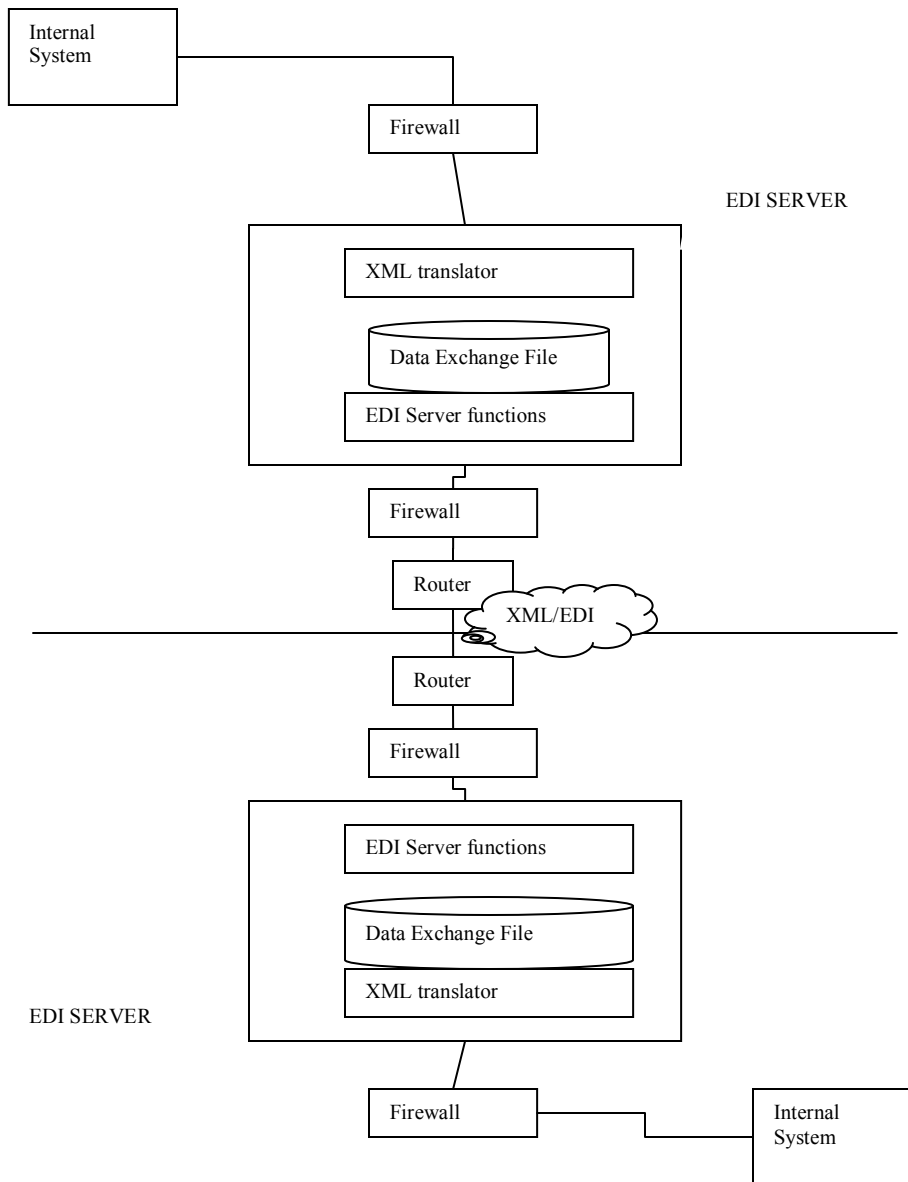


Figure 6: Direct connection of two EDI Servers

Figure 6 shows how two EDI servers can communicate using XML/EDI. The EDI server contains the XML schema and DTD to perform the XML conversion in the XML translator. The data exchange file contains XML data in the exchange area between the internal system and the EDI business document.

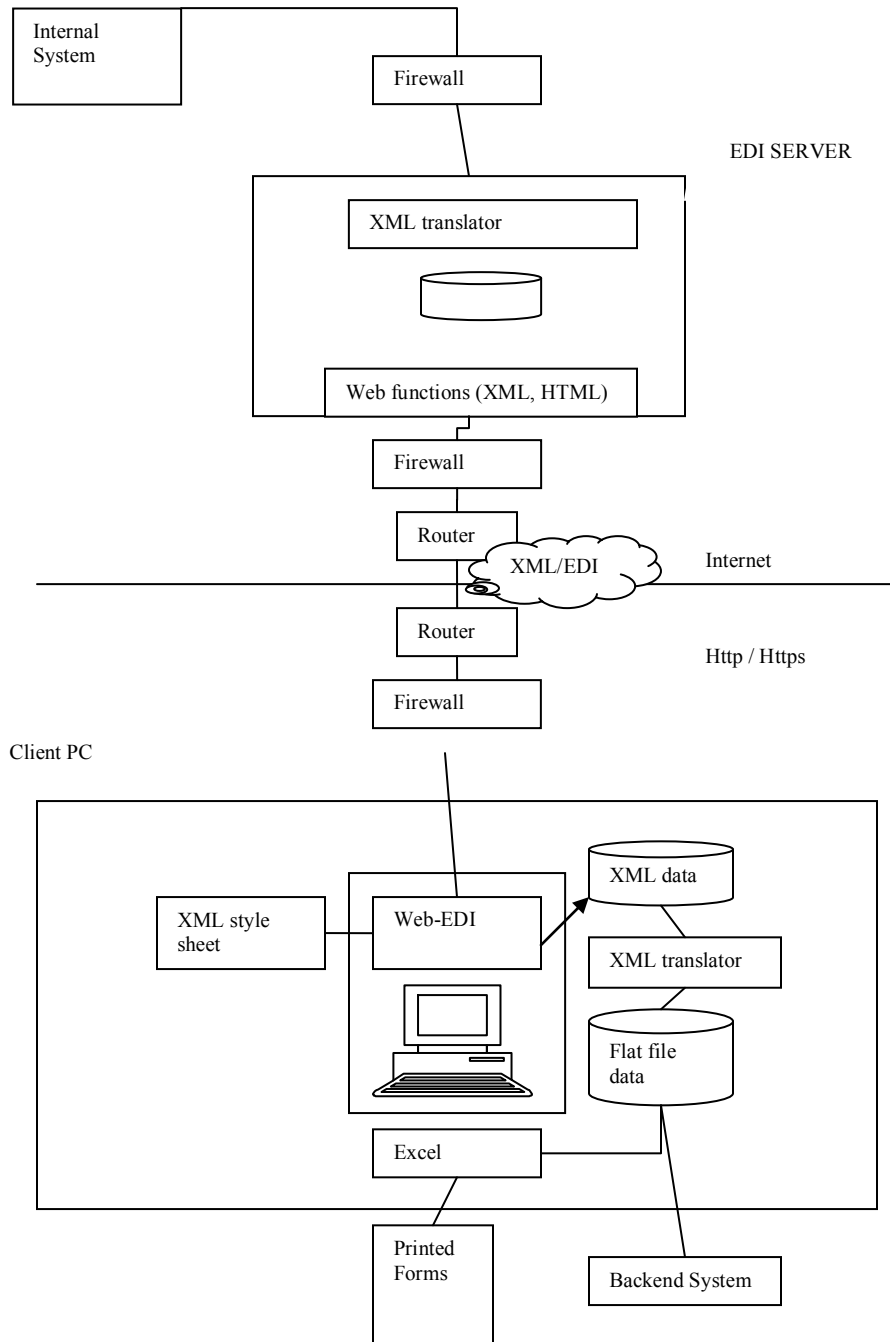


Figure 7: Direct connection scheme between client and EDI server

Server has some web functions that allow it to make documents visible to the client via the web. The corresponding Web-EDI function on the client side consists of a screen and data entry capabilities, the ability to submit a reply to the server and to download XML/EDI data. The translator generates CSV or flat file data. The client does not need to have any EDI capabilities in this setting.

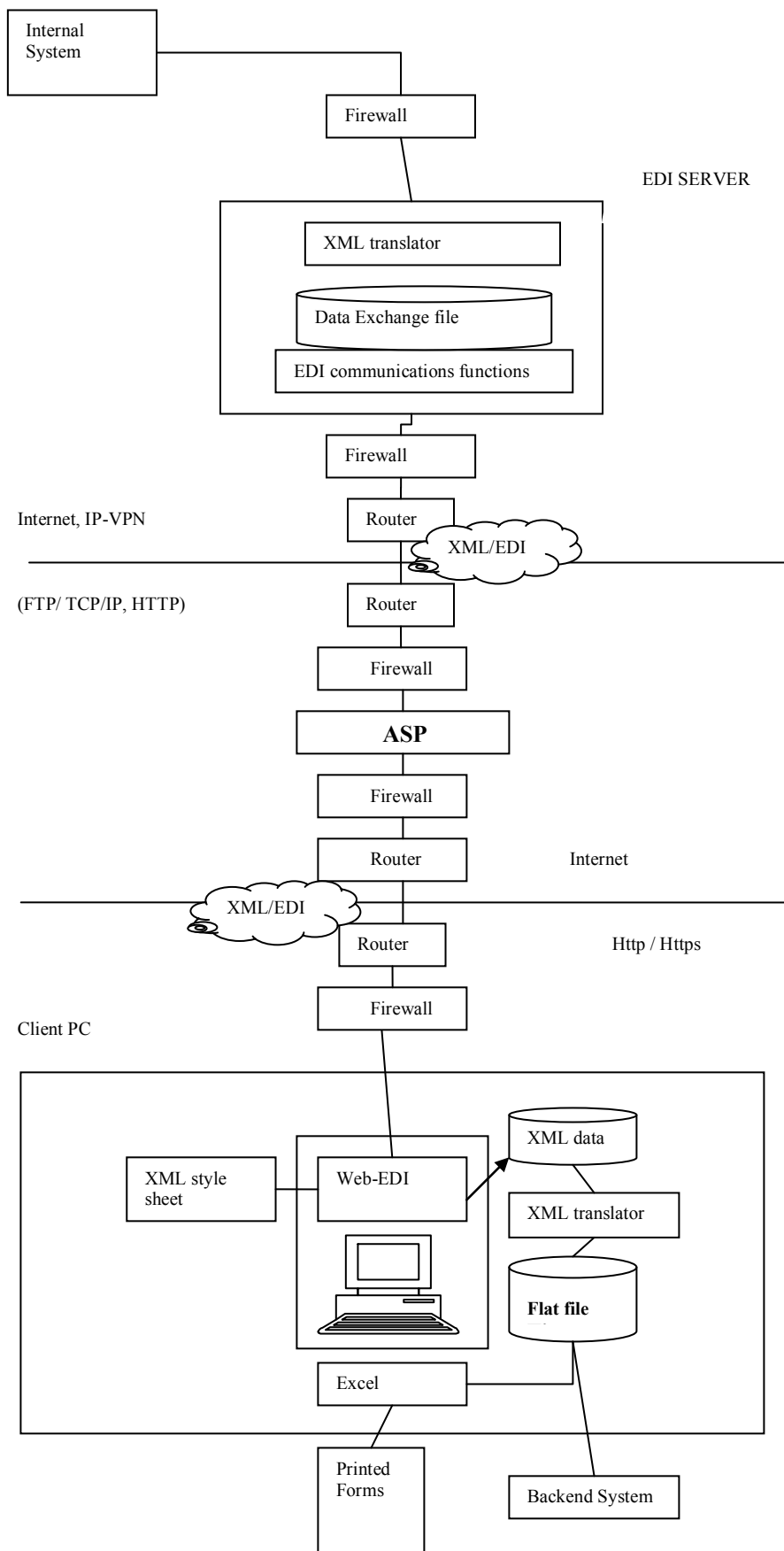


Figure 8: XML/EDI using ASP

In the setting of Figure 8 the XML translator function in the EDI server is provided by the ASP. The EDI communication module transmits files and keeps a history of transmissions. The ASP then performs the conversion so that the client can access the files through Web-EDI in the same manner as discussed before.

5.3 Collaboration XML/EDI

Collaboration EDI is based on the Internet and XML. It is targeted for real-time applications of business-to-business transactions in corporate backend systems. It is most successful in an environment where several business systems inside a company and many different systems used by customers, partners and suppliers must be connected in real time. If it is only used for business to business transactions and the companies operations force batch processing the benefits of collaboration XML/EDI are very small.

Traditional EDI focuses on the development and definition of business document message standards. Collaboration XML/EDI on the other hand defines and standardizes the business process not just individual messages.

RosettaNet[46] for example, develops universal standards for the global supply chain. RosettaNet's origins were in the need to trade complex information in high technology and adjacent industries, and now these efficient standards have spread to other sectors. In a dozen industries, spanning hundreds of companies, and the entire globe, the standards enable automation of business processes.

EbXML [19] is an attempt to internationally standardize collaborative XML/EDI. Its success has been at best mixed so far. Far more promising appear to be Web Services. A Web Service is defined by the W3C [59] as "a software system designed to support interoperable machine-to-machine interaction over a network". Web Services allow companies to connect and coordinate business applications over the Internet.

A collaboration XML/EDI system allows for the secure and reliable, real-time exchange of business documents on the message level on the Internet. This exchange is further coordinated with internal systems and uses the following processes:

- Real-time business information exchange
- Transmission and reception of single business documents in single messages
- Process management capabilities in business to business collaboration
- Coordination of collaboration with internal systems
- Creation, conversion , inspection of standardized XML business documents
- Security implementations such as electronic signature, encryption and authentication

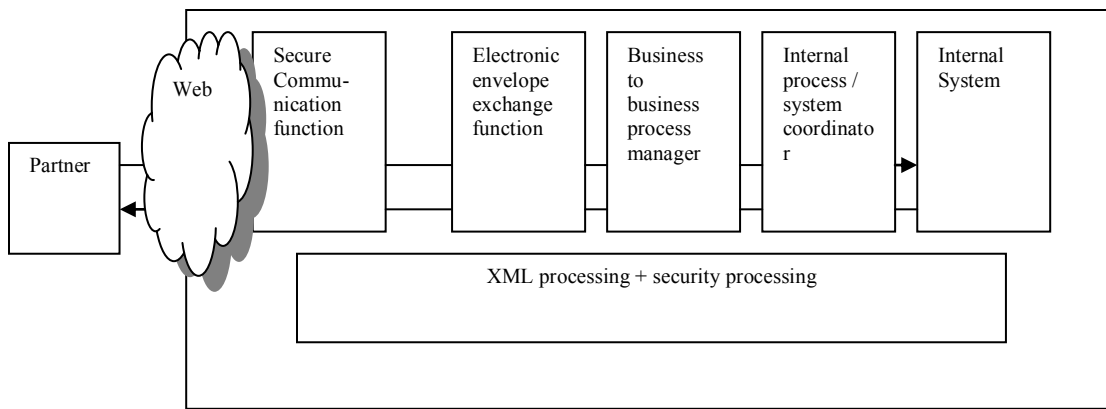


Figure 9: Collaboration EDI/XML Systems Architecture [65]

In a collaboration EDI/XML systems architecture (Fig 9) the secure communication module has responsibility for secure data communication. Security and authentication are ensured using SL or TLS. The module can securely send and receive business data. The electronic envelope module when sending receives an XML file from the B2B coordinator and puts this XML file into an electronic envelope. An electronic envelope here represents encryption, error correction and certification functions. Upon receiving of an envelope the functions are reversed. The envelope module also coordinates with XML processing and security processing.

The B2B manager is responsible for managing the currently applicable B2B communication standards and applying them to the given message or file. B2B communications are also called public processes while internal processes are called private processes. The Internal system coordinator fulfills a parallel role to the B2B manager with respect to internal communication. It coordinates internal processes with external (public) processes. The XML processing must ensure that the XML formats used are compliant with currently applicable standards. Finally the security processing module is responsible for all security related aspects of the communication such as digital signature creation, encryption/decryption and authentication.

6. XML conversion

XML documents contain only information about the structure of the given document and its content. The easiest way to read an XML document is to first convert it to HTML or XHTML and to then to display the resulting document in a browser. To coordinate between different systems the structure of an XML document must often be changed. An XML document that “describes” or “represents” an EDI file for example may not be usable in an internal data system – its structure must be converted or the document itself must be converted into another format.

There are four different types of XML conversion:

	Conversion pattern	Application
1	XML to XML	XML to XML linking e.g. conversion of an internal XML document to an XML document that describes an EDI file (XML/EDI).
2	XML to HTML/XHTML	Allows for easy readability of XML documents. (E.g. makes an XML/EDI document human readable through a web browser).
3	XML to other formats	XML system linked to a non XML system (e.g. an XML/EDI document is converted to CSV (comma separated value) format for easy further conversion to required internal data format).
4	Other formats to XML	A non XML system is linked to an XML system (e.g. an internal file in CSV format is converted to XML for use by an external EDI system).

Table 4: Types of XML conversion

6.1 XML conversion methods

There are four main conversion methods for XML documents:

A. Conversion using programs that were developed internally only for this purpose.

In this case a program is developed in a suitable programming language (C++, Java, Perl, C#, Python) that reads the XML document (or document in another format), analyzes its content and converts it to another format (to XML). A well written program will be able to handle all four conversion cases. This approach is successful with well trained and experienced developers. On the other hand programs developed in this manner are only able to convert between known XML document structures, changes in the document structure will require program changes and hence increase the maintenance costs. This is a good option for companies that have in house developers.

B. Conversion using Cascading Style Sheets (CSS).

In this case CSS that is used in html documents is applied to XML documents to allow for conversion. All common browsers are CSS compliant. The CSS conversion however can only designate the method of display (font, size, color, etc.) for the tag but the method cannot convert the structure of the document. The method only works for the second type of conversion (XML to HTML/XHTML).

C. Conversion using the eXtensible Style Sheet Language (XSL).

The conversion is performed by using the XSLT (XSL Transformations) developed by W3C and requires a XSLT processor. The XSLT processor is implemented on all platforms that can process XML. This method allows for conversion of the first three XML conversion types. It cannot be used to convert other formats to XML. The conversion definition document in XSLT is simply another XML document.

D. Conversion using commercial software.

There are many software products available on the market that allow for a conversion in all four cases. Before acquiring a software product it is essential for a company to closely examine its requirements. Once the package has been purchased no more changes can be made.

To decide which method to use it is necessary to closely examine the context in which the conversion should be employed. Note that for conversion from other formats to XML only custom designed programs or commercial products can be used. If the computing platform is fixed an in house developed program in a fixed programming language may be the cheapest option. If a conversion between operating systems needs to be performed (and to XML) XSL may be the cheapest option. Since EDI is used for business transactions a common operating systems and platform are unrealistic assumptions. With EDI the advantages of XSL appear to be tremendous. Many programmers are not yet experienced in XSL however. This currently greatly increases the cost of a conversion using XSL.

We next describe some examples and test results of programs that can perform XML conversions from X12 or EDIFACT to XML.

6.1.1 BOTS open source EDI Translator

BOTS [8] is an open source solution to EDI conversion. It is free to use with no licensing fees. It is provided by Ebbers Consult, Inc. The knowledge of python is extremely beneficial when working with BOTS. The entire program is written in python (open source, so source code available to modify) and additional user-defined functionality may be added by plug-ins written in python. The program is web-based, once installed on a server; any workstation can access it through a web browser. It supports most popular browser (IE7/Firefox/Opera/etc.). An understanding of networking concepts would be helpful for troubleshooting purposes. The program is still being maintained by the developer (latest update 2008-10-25, v1.4.0). It is highly configurable and EDIFACT and X12 grammars and plugins are available for conversion to XML (install and use). There is a decent manual and tutorials are available on the developer web page. Outside of that, there is a scarcity of online support. Users must pay for commercial support from the developers (Ebbers Consult, Inc.). It appears to be a one person enterprise, so a single point of failure. Supports Windows and Linux/Unix. The installation is relatively straightforward if running Windows XP or Linux as there are package installers that handle all the necessary requirements. In our experiments the program did not run under Windows Vista/7. The program is able to automatically receive EDI messages directly from the network that they are transmitted from. (VAN, internet, etc.). It can also perform batch operations. Because it is a web-based application, the GUI is very minimal.

6.1.2 m-e-c eagle

mec eagle[39] is an open source solution to EDI conversion. It is free to use with no licensing fees. It is provided by Mendelson e-commerce GMBH. The program is entirely written in Java, and is hence platform independent. It does not appear to be maintained by the developer anymore (latest update 2006-05-18). The only support available is a forum

on the company webpage and the forum is not active. There is no activity within the last year. No other online resources are apparently available. **Users** must pay the company for commercial support. The built in help files are extensive and easy to follow. The program installs and runs under Windows 98 and XP. It cannot be installed on Vista, but files may be copied over to run on Vista (running under compatibility mode) with some compatibility issues. It apparently does not function correct only Windows 7. The GUI is very user friendly and tasks and functions are nicely organized into visual functions. One is able to connect to VAN, Internet, E-mail, etc to automatically receive EDI messages. The program can perform batch operations

6.1.3 Microsoft Biztalk 2006

Biztalk 2006 R2 [5] includes support for EDI exchange (including X12, EDIFACT, and HIPAA support) and Availability Statement (AS2) data for EDI over the Internet. A base EDI adapter along with its components must also be installed. It is **commercial** software. The Software includes many other E-commerce features (ex. RFID, Business Tracking). It is powerful enough for use by large companies with extensive e-commerce capabilities. It can be overwhelming for clients that are not interested in all the other features. It includes extensive support (Official support from Microsoft via MSDN). It also includes a detailed EDI to XML conversion tutorial. It has official support from BizTalk developer teams through MSDN blogs, specifically, an official blog from the BizTalk 2009 – EDI product Team that is dedicated to all EDI news on BizTalk, with posts from the developers. There are active online community forums to ask general BizTalk questions or specific EDI issues within Biztalk. It is a very large program relative to the other solutions (because it covers more than just EDI and XML). Software and licensing are expensive.

6.2 Conversion methods for XML style sheets

XSL is a framework to describe the layout of an XML document; it consists of the XSLT (XSL Transformations), XPath and XSL FO (Formatting Objects) specifications. An XML style sheet or XSLT style sheet describes the format conversion formula compliant with XSLT.

1. XSLT (XSL Transformations)

XSLT is the language of style sheet descriptions. In the style sheet the conversion of an XML document is described. For example,

```
<xsl:element name="Element">
  TEXT
</xsl:element>
```

is converted to

```
<ELEMENT>
  TEXT
</ELEMENT>
```

in XML.

2. XPATH

XPATH is a language used to specify parts of an XML document. E.G.

```
/child::doc/child::chapter[position()=5]/child::section[position()=2]
```

selects the second section of the fifth chapter of the doc document element.

3. XSL-FO

XSL-FO is an XML-based markup language describing the formatting of XML data for output to screen, paper or other media.

6.2.1 XSL Conversion

In an XSL conversion an XML document, using an XSLT processor and an XML style sheet is converted to an XML document, HTML document or an XSL-FO document. To display an XML document on the screen, i.e. in a browser window it is first converted to an HTML document. This is the method used by Web-EDI systems to display a document on the screen.

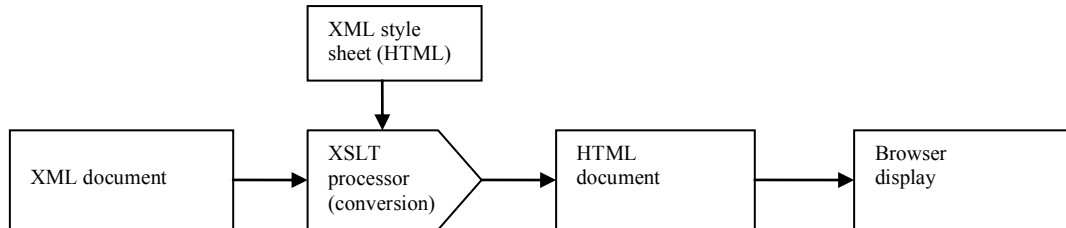


Figure 10: XSL conversion for screen display

To print a document using the XSL style sheet, an XSL style sheet is written for conversion to an XML-FO document. The XSL-FO processor then formats this document to print it or to output it as a PDF file.

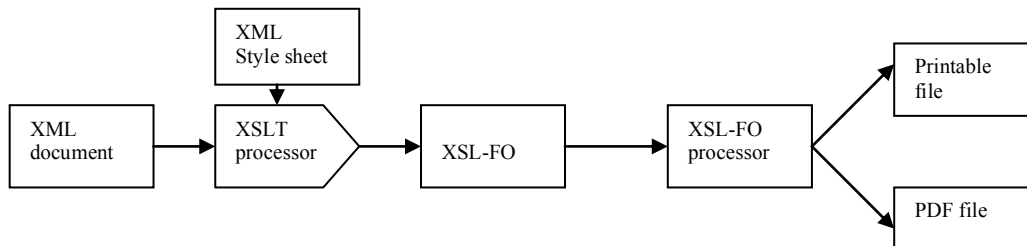


Figure 11: XSL conversion for printing

Advantages and disadvantages of adopting an XML style sheet

(A) Advantages

- (i) XML style sheets are standardized. Moreover the standard is an open standard and not proprietary. This allows vendors to develop applications that work on many different platforms.
- (ii) Style sheets enable the separation of structure from content. Namely the content is provided in XML while the structure is given in XSL. Assume company A is connected via XML with company B and company C and B and C have different order formats. So company A must be able to handle order forms from both B and C. If structure is separated from content, company A can use the same XML document for both company B and C – only the XSL form needs to be changed. This facilitates management and integration with the backend system/database.

(iii) Style sheets provide for a very high level of expression. With XSL-FO we can either display one and the same file or print it.

(B) Disadvantages

- (i) The style sheet requires an investment in the training of programmers. This cost may be reduced with new XSLT tools.
- (ii) There are currently only a few products compliant with XSL-FO.

This clearly implies that the definition of style sheets is the crucial step for the effectiveness of XML based business to business communication.

Ideally style sheets could be stored on a server that is accessible by all participants. Companies could then simply download the style sheets they would like to use for their business applications. This facilitates the exchange of common style sheets between business partners. In another approach companies would exchange style sheets in a bilateral manner. This model is closest to the current state of affairs in EDI communication.

The efforts to standardize XML/EDI are moving in two directions. First the standardization of the definition of basic business documents and data elements and second the standardization of message transmission and security.

Because of the extensibility of XML it is possible to standardize any desired industry standard business documents. There are several available methods to convert standard business documents into XML. Most commonly the new XML standards are simply based on some already existing business document standards.

6.3 XML-EDI Conversion Experiments

In our experiments we developed and tested simple EDI to XML conversion tools. The goal of these tools is it to convert EDI data into XML so that it can be integrated more easily into back office systems and be made available through the Internet to others. In general for an experienced programmer it is not difficult to develop such a tool. In addition there are free tools available online that can be modified to a users needs. In particular the language Python has become popular with developers that work on EDI to XML conversions. Python is an object-oriented, byte-compiled language with a clean syntax, clear and consistent philosophy, and a strong user support community. Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones. Python has also been ported to the Java and .NET virtual machines. Python is distributed under an Open Source Initiative (OSI) -approved open source license that makes it free to use, even for commercial products. This makes it possible to quickly write working, maintainable code, which in turn makes Python an excellent choice for many programming tasks. Processing any type of EDI is no exception.

Our experiments showed that such a tool, however, is likely simply an equally cumbersome extension of EDI. Namely the tool must know in advance the type of EDI it is supposed to convert. That means that potentially for each separate EDI standard a separate program is needed. Hence even if one can develop a sufficient number of conversion tools that would allow a company to use these tools to convert standard EDI documents to XML, the tools would not be able to handle all exceptions. So if an unforeseen situation arises the company would face the same difficulties with these conversion tools as it did with EDI. Also since Python is open source, while one can find

excellent help from the Python community, no specific vendor will be available to help a developer solve problems.

The in-house conversion approach requires technically advanced individuals, which a company must take into account when considering appropriate staffing. Finally, the framework we developed is not mature enough for production use and other similar freely available frameworks may not be overly mature, either.

7. EDIINT – EDI over the Internet

EDIINT – EDI over the Internet – is a working group of the Internet Engineering Task Force (IETF) enabling the transport of EDI and XML data over the Internet in a secure manner. It is also the standard of an alternative data transport to value added network (VAN) based data communications between EDI trading partners. The service provide matches the services associated with VAN services – mail boxing, trading relationship management, security, authentication and non-repudiation – via software[30].

EDIINT includes three major standards: AS1 (Applicability Statement 1 for SMTP protocol, asynchronous – “Batch Mode”), AS2 (Applicability Statement 2 for HTTP protocol, synchronous – “Peer-to-Peer, Real Time”), and AS3 (Applicability Statement 3 for FTP protocol, synchronous – “Client/Server”). The AS1 uses email attachments with S/MIME (Secure/Multipurpose Internet Mail Extensions) encryption and security over SMTP (Simple Mail Transfer Protocol); AS2 provides S/MIME encryption over HTTP (Hyper Text Transfer Protocol); while AS3 provides S/MIME encryption over FTP (File Transfer Protocol) in a server/client model manner.

Several major retailers and large manufacturers are in the process of implementing or have already implemented EDIINT initiatives that will eventually require their supply chain to make the change to eliminate VAN costs. Wal-Mart, the world’s largest retailer – which never used VAN’s - has moved to EDIINT replacing their “bisync direct dial communications” and is now requesting for all their vendors to accommodate and follow. Several other companies followed Wal-Mart’s initiative. As a result many major software companies provide software and services that support EDIINT/AS2.

EDI started in the 1960s as a bisynchronous communications protocols of the IBM mainframe environments. In the 70s, EDI added store and forward networking that became the predominant model used by Value Added Network (VAN) service providers since then. VAN’s provide third party, auditable features guaranteeing a reliable and secure exchange of electronic business documents between businesses. The mail boxing services VANs provide allows each trading partner to process data on their own schedule. If companies use a VAN the cost depends directly on the number of documents and characters exchanged. Even though VAN providers have lowered their per document and per character charges the number of documents in general has gone up increasing the VAN costs for many companies. In the last two decades internet bandwidth and ease of access have increased steadily. Hence the Internet appeared as an obvious candidate for a more cost effective alternative to VAN’s. After some security, authentication and non-repudiation issues had been resolved the Internet Engineering Task Force began to

address standardization and added mail boxing and trading relationship management to the package.

Even though traditional EDI is standardized, standards are changing at a very fast pace since competing companies in several industries operate with specialized business arrangements, which may only last for a few months or few transactions.

In most EDI transactions using VAN's, negotiated Trading Partner Agreements (TPAs) are used to specify data interchange on a one-to-one basis. The case of Wal-Mart mentioned earlier is not an isolated case however. Namely on the Internet and in electronic commerce, there is a trend toward what one could call Unilateral TPAs [30]. In a unilateral TPA one party specifies the standard to be used for a transaction and asks any potential business partner to submit transactions of that type. Another benefit of Internet EDI and EDIINT in particular are the real time capabilities. While traditional EDI provides a batch driven process with a wait time between submitting and receiving a confirmation for a submitted document, EDIINT exchange happens in real time. The AS2 and AS3 protocols specify how to deliver a document to the recipient with no intermediate routing or mail boxing. An AS2 and AS3 capable application at the sender establishes a connection over the Internet to the receiver's application and sends the document. The receiver then returns a receipt to the sending system.

7.1 EDIINT Security

AS2 provides several security options. Data can be sent over a secure connection (HTTPS) or package encryption (using a digital certificate for authentication) can be used. AS2 also allows for documents to be digitally signed, making it possible to later check the validity of a document. Other aspects of AS2 may be challenging for some smaller users namely smaller companies. First it requires a company to have at least one computer connected to the Internet at all times. In particular, a requirement of EDIINT AS2 is the Internet visibility of a valid IP address.

AS1 uses email attachments and is routed through existing email servers and protocols. AS2 on the other hand requires from a company to open up a portion of its enterprise network to the outside world to enable the receipt of files via HTTP/HTTPS. In some cases it may be challenging to get network administrators on board.

Second, AS2 requires the management of digital certificates. These certificates can be revoked at any time and expire periodically. Hence someone must visit all issuing authorities and look at the revocation lists. Also expired certificates must be renewed. Clearly the amount of work will grow with each new business or trading partner connected via EDIINT. The AS2 application requires the manual import of Digital Certificates and there are no commonly used standards to deal with revocation.

AS3 is a new MIME based protocol specification from the Internet Engineering Task Force. AS3 defines how to perform secure and reliable file transfers with FTP in a standardized way to ensure interoperability between solutions. Classical FTP has virtually no security and reliability features so most firewalls will not allow FTP files to pass by anymore. Hence AS3 adds security and reliability features to FTP. No repudiation is also ensured since encryption is document based and does not interfere

with individual network packets. As a result it is much more compatible with firewalls than Secure FTP using SSL.

While AS2 is a peer-to-peer model, AS3 is a client-server model. It uses receipt notifications like AS2 and unlike AS2 does not require a “listener” to always be connected to the Internet.

It is also suited for Dial- up Internet connections, making it an option for partners with very limited internet connectivity.

We next provide several brief examples of EDIINT AS2 compliant software solutions. All solutions are stand alone products. The manufacturers claim that they can be plugged into an existing EDI infrastructure without significant changes to current operational set ups.

7.1.1 bTrade TDAccess/TDPeer

TDAccess/TDPeer [11] is a real-time communication platform designed for smaller trading communities. It includes support for at least 15 EDI translators, task scheduling and integration into backend office systems. It also provides AS1, AS2, VAN/SSL and PKI certificate security.

The TDPeer platform has three components. The client has a command line DOS or Windows interface. An AS2 server is the http listener needed for real-time capabilities. Trading partner relationships and certificates can be set up with the TDManager module.

7.1.2 Inovis BizConnect

BizConnect [31] is a Java-based data exchange solution designed for small-to-medium size enterprises with up to 25 business partners. BizConnect comes with pre-configured partners (Ace Hardware, Auto Zone, Wal-Mart, etc.

7.1.3 Gentran Integration Suite

Sterling Commerce is one of the world’s largest providers of EAI and B2B software solutions.

Sterling's Gentran Integration Suite [25] can handle both small transactional messages and large bulk data and batches of messages.

The Gentran Integration Suite supports EDIINT AS1 and AS2. It also features mailbox services, real-time and batch processing, and is suitable for large trading communities.

Sterling's mailbox functions are similar to mailboxes in a mail server:

- Each user can have one or more mailboxes
- Mailboxes can be role based instead of user based
- A user can be continually connected to the mailbox and process messages continuously
- Users can leave message in the mailbox and then process them on mass
- Rules can be applied to the messages as they come in to a mailbox to determine how they should be handled.

Since everything goes through a mailbox the applications do not have to be aware of the differing partners.

7.2. Problems with EDIINT (and Web-EDI)

To begin using Web-EDI a client needs little more than Internet access. This led to a fast growth in the number of Web-EDI systems. These early systems were mostly based on html however and so the screen display formats were not standardized. This means that for a company with many different trading partners the system must be able to understand the traded items on the screens of each partner. To coordinate with in house systems conversion is again necessary.

To solve this problem EDI standard business documents and screen formats per industry must be developed. As a result, however users lose the ability to customize screen formats for their own needs. Also the problem for cross industry trade remains. XML provides a convenient solution to these problems. Screen formats can be written with XML style sheets and provided to each client. The client can then modify the style sheet to customize the format for his own needs while at the same time being compliant with standardized EDI documents.

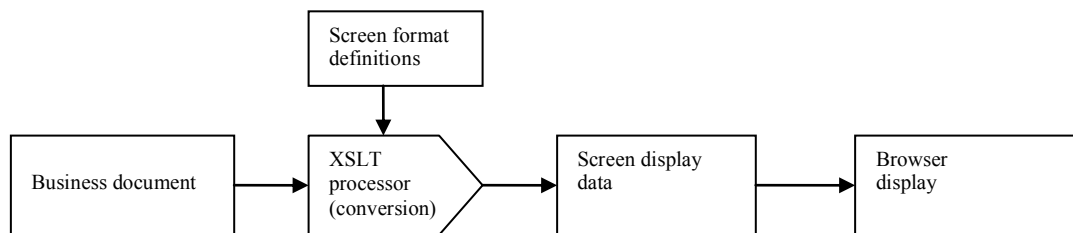


Figure 12: XSLT: XSL transformations

In Web-EDI using XML the conversion by the XSLT engine using XML style sheets can be done either on the client or on the server side.

In case of **conversion on the server side** the business document (in XML format) is converted to HTML (XHTML) by the server and then transmitted to the client. From the client's perspective this is simply HTML based Web-EDI. The conversion itself can either be done each time a request is received (in-time-conversion) or once after the business document is created (ahead-of-time-conversion). In the case of in-time-conversion business document data can be exchanged dynamically with the client at the cost of additional overhead for each conversion. Ahead-of-time-conversion incurs minimal overhead but leads to static data.

Conversion on the server side leads to a lack of flexibility for the client side. The client cannot influence the screen format and may have difficulty adjusting the received data to the client's backend system. On the other hand the server can implement all security requirements and assure security and authenticity (using certificates). Conversion on the server side also does not depend on the client's browser. It works even with browsers that lack an XSLT processor.

Conversion on the client side and XML style sheet management on the server side

The server stores business document data (XML) and the XSL style sheet. Both are sent to the client on request. With the received data the client creates an HTML (XHTML) document

through XSLT processing. This approach allows the server to manage multiple style sheets for many different clients. Also the client does not need programmers to develop the XSLT style sheet and to monitor the XSLT processor. All newer browsers have an XSLT processor built in (e.g. MS-XML in case of Internet Explorer).

Conversion on the client side and XML style sheet management on the client side

The server stores business document data (XML) and submits it to the client on request. The client develops his own XSL style sheet and uses it to convert the data on receipt to HTML (XHTML). In this approach it is suggested that the server provides an XML style sheet template to the client. The client can then modify it based on his needs. The approach gives maximum flexibility to the client. The client has the ability to customize the display of the file and at the same time does not compromise his ability to ensure interoperability with his backend system. This approach however requires employees that are familiar with XSL style sheets and XSLT processors. The XSLT processors must be implemented on the client side.

Data transfer from the client to the server

There are two options to implement data transfer from the client to the server. In the first option data is entered manually by the client in an HTML web page. In this case the data entered does not remain on the client side so the client will have no direct record of his transaction. In the second option the client runs a processing program that accepts business document data, converts it to XML and sends it to the server. In this option a copy of the data sent remains on the client side. The development of the conversion program however incurs additional costs.

Many companies are hesitant about transmitting their business data over the Internet with obvious concerns about security and reliability. To address these concerns the use of VPN's (Virtual Private Networks) has been suggested. In particular IP-VPN's (Internet Protocol) are available from many vendors and are known for their outstanding security and excellent performance. A VPN is a virtual private network created on public networks. It ensures security similar to dedicated networks using tunneling and related technologies. In tunneling to enforce security an extra header is added to each packet.

- Internet VPN – A virtual private network built on the Internet. It is commonly used in corporate intranets and extranets between companies.
- IP-VPN – In this case the virtual private network is built on a dedicated, closed network provided by a network service provider. The communication protocol used is based on IP. Used as corporate intranet and extranets between trading partners.

8. The Language M

M is based on XML and aims to address the interoperability issues of XML. It was created at the MIT Data Center [10] as an open, global language that communicates between proprietary schemas enabling companies to combine, visualize and understand data. Like a regular spoken language. M has a dictionary to describe the meaning of words. The dictionary consists of a collection of definitions that can be used when making computer transactions. The dictionary of M also includes word relations, data format, and language translations. These all help to form and understand messages written in M.

Just like any other human language M has rules to organize words for machine understanding of messages. Initially, there were three simple rules namely phrases, key-

value pairs, and tables. A phrase represents a sequence of machine-understandable words that has a single meaning. Phrases are essential in M in a sense that they help to describe data elements more precisely. Key-value pairs are lists of words that have associated data values. They may be used to represent tax forms, medical records and financial statements.

Key value pairs also help to make data interoperable within a source as well as from outside sources. Interoperable data helps to combine data on the Internet with the internal data of a company. And finally, tables are the most common way to store data.

Since M helps data and models flow freely across the network, it has many useful applications. Namely it can be used as an intermediary between proprietary data systems. Data from one database is translated into M before it reaches the other database. The following table shows this transfer of data.

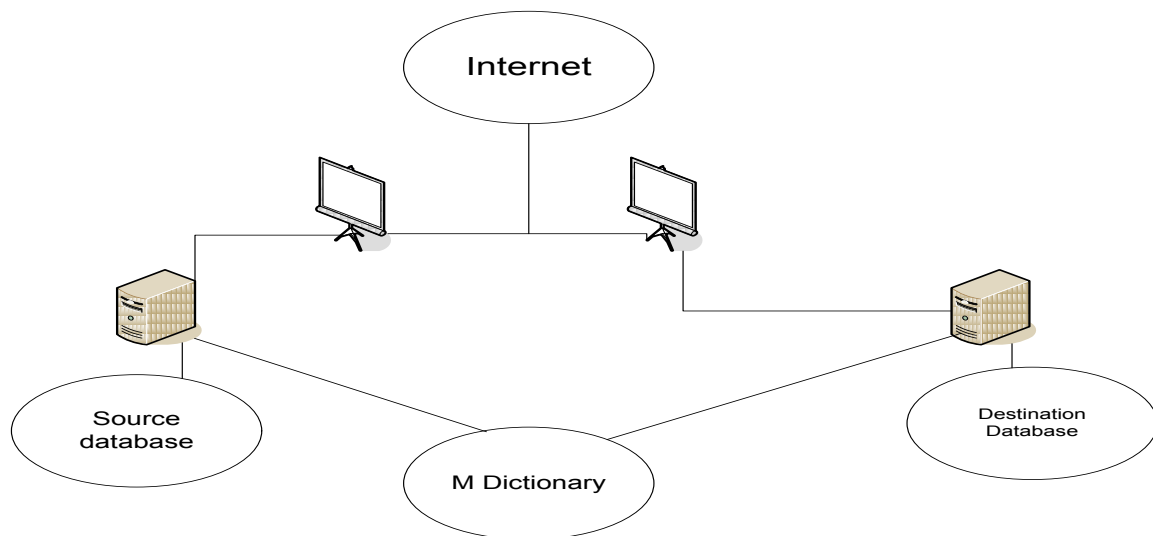


Figure 13: Communication structure using M

1. Beginning at the source database, data is retrieved.
2. The data is transformed into M using the M dictionary
3. Data is sent from the source to the destination over the Internet
4. Data is transformed back into the destination database format.

8.1 Comparing M and XML

Like M, XML may describe data transferred between systems. But, its tags are not as readable as words in the M language. The tags may become verbose and redundant, making it even harder to read. For example, a tag could be defined to be “<ContainersPerDay>”. Also, this tag may not have the same meaning across different systems. So in a supply chain with a lot of raw data distributed across different systems in different locations, this lack of consistent meaning causes interoperability problems.

As mentioned before, to help standardize communication, XML requires a schema to define the tags used in an XML file. But, many such schemas (e.g. XHTML [63], SOAP [9], SMIL [53], ebXML [19], RosettaNet [46]) exist, complicating communication. Thus, unless a separate translation using say XSLT is performed, a conflict may occur when a source database uses one XML schema to translate for transmission and a destination

database uses another schema. With M, a single dictionary avoids this confusion. Centralizing the meaning of the metadata in transmission helps M avoid the problem in XML where a separate translation between each pair of schemes needs to be performed. M and XML may be mixed together, with M words used as the tags in XML. While M appears to solve many of the problems that XML has, it was just recently released as a prototype and is not yet ready for commercial applications.

9. Communication Model at the Los Angeles and Long Beach Ports

We distinguish the following three communication models:

- a. **The Bilateral Information Model (BIM).** Here information is exchanged directly between the interested parties in a bilateral manner. Most large companies at the ports currently use this model.
- b. **The Centralized Information Model (CIM).** Here information is stored in a central database. Agents that have the right to do so can retrieve information from this database
- c. **The Decentralized Information Model (DIM).** Here data is stored and controlled by each individual agent that owns the data. A broker service can help an authorized agent to find and retrieve data.



Figure 14: Current use of EDI at the ports of LA and LB

The bilateral model (a) is currently prevalent at the ports. There are strong indications that this will remain the case for the foreseeable future since it is unlikely that terminal operators will be willing to store all their business data in a central database that they share with other terminal operators. EDI is used between terminal operators and rail companies and separately between terminal operators and shipping lines / carriers (Figure 14).

9.1 Terminal Operating Systems – Navis Sparcs N4

Currently many terminal operators worldwide are in the process of installing (or have already installed) sophisticated and state of the art Terminal Operating Systems (TOS). The APM terminal, for example at the Long Beach port is as of February 2009 in the process of upgrading to a new version of the Navis Sparcs TOS. Navis Sparcs N4 is a TOS that promises to deliver the “most scalable, open, deployable adaptable and maintainable TOS available” [42].

The biggest cost factor when installing a TOS for many companies is the customization of the off the shelf system. While the TOS may be able to solve many problems a terminal faces it may not immediately be able to solve the problems that this particular terminal faces. This leads to additional costs that may not be predictable at the outset. Because of this Navis developed a TOS that is designed and promoted to be easily customizable after it has been installed. Navis uses what it calls an extensive data driven business logic to create this highly configurable solution. Users are for example able to define fields and to configure hold and permission rules to tailor the system to their business operations without having to write code. The TOS can also be extended through software development kits and application programming interfaces (APIs) so that a user (a terminal) can connect its TOS with other best of breed systems.

N4 is a fully integrated system from gate to yard to vessel, eliminating the need to coordinate between several disjoint systems. N4 also allows for the integration of several facilities so that a large company can run it not only to administer an individual terminal but also use it to integrate all its terminals together, further reducing integration costs. The user interface is based on a Java Rich Internet Application (RIA) and as a result looks immediately familiar to today's users who grew up with Windows. For most users of this TOS there is no need to be familiar with command line operations, greatly lowering the learning curve.

Navis even claims that new EDI partners and messages can be quickly configured by IT administrators without writing any code. The system provides API's to connect with other critical applications. Users keep the ability to decide whether to use internal IT development or third party consultants or outsourcing or Navis to provide these applications.

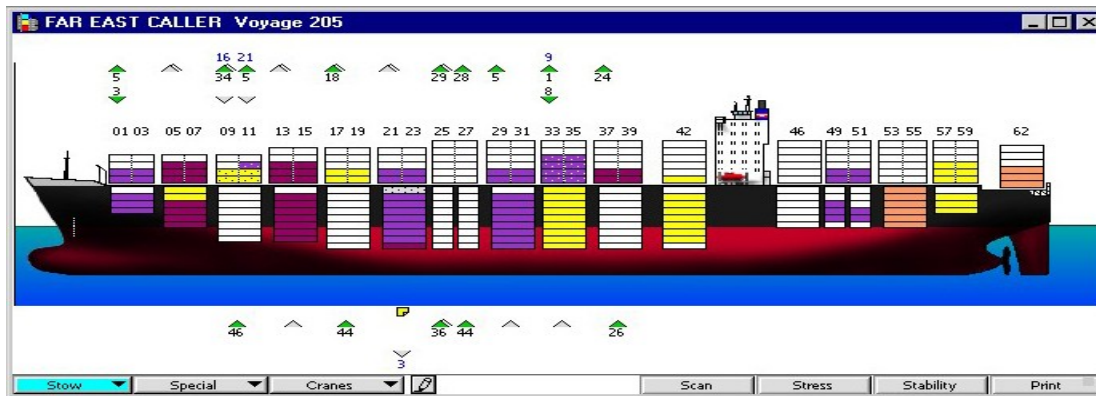


Figure 15: Navis Screen shot showing vessel profile [42]

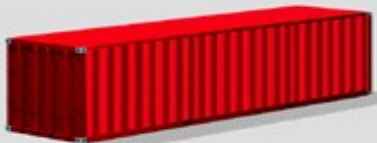
YOK 4200 ■ 19	Equipment ID: TPUH5340117 Category: THROUGH Status: FCL Group Code: Book/Rlse: Chassis:	 40' x 8'06" 19.4 mt General Purpose (4200)
I/B Vessel: FAREAST (FAR EAST CALLER) POL: Seattle O/B Vessel: FAREAST (FAR EAST CALLER) POD: Yokohama Line Operator: AC Destination: YOK		
Commodity: Stow Factor: S/YOK/4000/DRY Stacking Factor: S/DRY		
<i>position</i> => FAREAST-580382		
Time of Update: JAN/98		

Figure 16: Navis Screen shot showing Container information [42]

Figure 15 and 16 show two screen shots from the Navis TOS. Figure 15 shows a vessel profile while 16 shows information about a container such as origin, destination, type and storage location.

Navis SPARCS N4 product line is based on Canoo's UltraLightClient (ULC) [12], a 100% pure Java solution for Rich Internet Applications. With this pure Java technology in place, how can Navis achieve the flexibility it claims?

When creating multi-client container terminal software it is obviously not enough to create a solution that is stable and works for one particular client. The solution must be prepared for the numerous requirements that differ from client to client. One solution would be to maintain slightly different versions of the product for each client but this can quickly lead to a maintenance nightmare. Instead, the approach Navis followed is to include *scripting capabilities* (allow users to write programs that control other applications) into their product such that the client or the technical on-site consultant can apply the required adaptation. Since any adaptation has to express logic like special routing rules for containers, a simple configuration file is not enough. One needs a smart configuration, which is a typical usage pattern for Groovy [27].

9.1.1 Groovy

Groovy [27],

- is an agile and dynamic language for the Java Virtual Machine
- has the strengths of Java and in addition features inspired by languages like Python, Ruby and Smalltalk
- makes the newest programming features available to Java developers with a very small learning curve
- supports Domain-Specific Languages (programming languages or specification languages dedicated to a particular problem domain, representation or solution) and other compact syntax so code becomes easy to read and maintain
- makes writing shell and build scripts (programs that control other applications) easy with its powerful processing primitives, Object Oriented abilities and a software tool that automates software building processes for Domain Specific Languages (ANT DSL).

- potentially increases developer productivity by reducing scaffolding code when developing web, graphical user interface (GUI), database or console applications
- simplifies testing by supporting unit testing and mocking out-of-the-box
- seamlessly integrates with all existing Java objects and libraries
- compiles straight to Java byte code so users can use it anywhere they can use Java

Groovy is very well suited for Java-based RIA solutions since it integrates seamlessly with Java while providing higher flexibility and expressiveness where needed.

Groovy uses an AJAX (Asynchronous Java Script and XML) framework. But it hides the implementation details of the AJAX framework in use. So if a user changes the AJAX framework in the middle of his project, the Groovy application code stays the same.

RIA and Groovy can be found in technology-leading, demanding, agile contexts where high requirements for user interaction design have to be met. While the RIA technology helps with presentation enhancement, Groovy can keep workflows, business rules, and domain models easy to modify on the fly. This is where Groovy shines and in the case of Navis Sparcs N4 enables users to customize their TOS to their particular needs.

9.1.2 Announced features of Navis Sparcs N4

The following is a list of features announced for Navis Sparcs N4:

1. Optimization of equipment work assignments in real time by pooling equipment across cranes and combining yard and equipment constraints with operating business rules.
2. Optimization of yard management: Expert Decking assigns each container its optimal position based on yard constraints and business rules.
3. Optimization of vessel planning: Auto Stow feature combines stowage factors (type, weight) with operating strategy and yard constraints in real time to improve vessel stow plans. Navis claims a reduction of up to 70% in planning time.
4. Quay commander: Real time monitoring of crane and vessel activities, vessel container moves and vessel labor assignments to allow for dynamic adjustments to vessel load/discharge times and crane sequences.
5. EDI: System has an interactive transformation designer that allows users to create maps for new message types. EDI files can be received or sent via ftp, email or Web Services. Includes multi stage conversion, processing and transmission tools. Supports gate, yard, vessel and rail operations.

N4 has a multi-tiered architecture based on J2EE servers and industry standard databases. It uses clusters of database and application servers to provide fault-tolerance, redundancy, load balancing and failover support (in case of a hardware or network failure). All users see the same data and work orders, updates are communicated throughout the entire system in real-time. The system also tracks the real time productivity of container handling equipment (CHE) and automatically dispatches work orders to radio data terminals to increase truck driver productivity.

10. Navis Sparcs and Web Services

In a white paper [55] Navis suggests to use Web Services to create a tighter integration between truckers, railroads, shipping lines, shippers and terminal operators. A Web service [59] is defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network". While EDI once correctly implemented is very well suited for document exchange it was not designed for process integration. What is the difference between document exchange and process integration? Consider, for example, an EDI COPRAR message [14]. This is a message sent by an ocean carrier to a terminal to specify which containers should be loaded onto a specified ship. The message is a well established standard and contains all the information needed to specify the containers and the ship they are supposed to be loaded on. Let us assume a terminal receives such a COPRAR message instructing it to load a container onto a ship that was already earlier loaded onto another ship. What should the terminal do? Discharge the container and reload it or leave it on the first ship? Who pays if they have to discharge it? Maybe it will not be possible for the terminal to get an answer right away from the shipping line so it will have to make a unilateral decision.

The COPRAR message will not tell the terminal what to do and there are also no other standardized EDI messages that could provide an answer. In this sense EDI messages do not implement processes i.e. a sequence of steps that can be followed to solve a given problem but instead simply exchange documents. This can lead to service failures, higher costs and frustrated terminal customers such as shippers.

What are the alternatives to the EDI message exchange? Ideally the logistics agent at the shipping line would like to be able to access his/her computer to find containers to reroute from one ship to another. The computer could then send a message in real time to the terminals computer system. The terminals computer system would then indicate whether it is possible or not to move the container and if it is possible at what price. The agent then sees the options on his screen and selects how to proceed. This sequence of steps is a process [55]:

1. The shipping line requests rerouting for a set of containers.
2. The terminal replies that either
 - The container can be rerouted.
 - The container cannot be rerouted.
 - The container can be rerouted at a cost of \$
 - The container was not found.
3. The shipping line confirms or cancels the rerouting of the containers.
4. Terminal confirms completion of rerouting.
5. Shipping line updates computer system for containers with new route.

This requires an integration of the process between the shipping line and the terminal. Some steps are done by each of the two computer systems. Both parties learn about the outcome simultaneously.

To implement such a process integration **real-time communication** between computer systems is needed.

EDI was designed when wide area networks were very slow and unreliable and before the time the Internet became today's ultra fast worldwide communication enabler. At the time when EDI was developed the above process could not have been integrated since it would have taken too long for the required messages to be forwarded and processed. Web Services in this sense are a new messaging scheme that requires low latency (real-time) communication to achieve integration between disparate, autonomous systems [55].

Why not instead 'fix' traditional EDI to implement the same thing?

Because of the enormous amount of network bandwidth available on the Internet traditional EDI is not able to compete with systems that have real-time capabilities. Setting up EDI communications is still a time consuming and expensive process. A company with multiple trading partners must go through this cumbersome process for each partner. This explains why outside the Fortune 500 only very few companies voluntarily decide to implement EDI. For most of the medium and smaller size companies EDI is simply a business expense that allows them to be in business with one of these larger companies that require the use of EDI. Otherwise these companies would very likely not choose EDI [36].

What makes EDI so difficult to implement and expensive? Because there is no standard process for setting up EDI, every setup requires lengthy meetings between partners to decide which messages to use, how to handle exceptions etc. Both parties then must implement the messages and coordinate with the other to test the messages. Commonly the partners will find flaws or errors in each others implementation and will then have to meet again to decide who should fix these flaws. They also need to decide which transportation medium to use – Internet, Value Added network or leased line. In general the setup process alone usually takes several months making it inefficient and expensive.

Web Services on the other hand *decouple* users. With Web Service Technologies and the Web Service Description Language (WSDL) [61] in particular the properties of the *technical interfaces* of a Web Service can be described as an XML document. A user that wants to *consume* a service that is offered by another user does not need to know anything about the implementation of this service. All the user needs to know is the interface of the service, i.e. how to interact with it. The Web Service Technologies and WSDL, however, do not provide support for the complete life cycle of a Web Service. Namely they do not publish or help to discover or communicate the details of the invocation sequence or address security and monitoring requirements.

To be useful in the context of B2B communications it is essential that Web Services that companies offer to each other are easily discoverable. To discover a Web Service there are three main approaches [37]:

- UDDI (Universal Description, Discovery, and Integration) [57] is a standard for centralized repositories. The first UDDI Business Registry (UBR) nodes were run by IBM, Microsoft, SAP and NTT Com.
- Service directories (or portals) which gather services using focused crawlers or by manual registration and offer search functionality via a HTML interface.
- Standard Web search engines which are able to restrict the search in some way to retrieve WSDL descriptions. This provides no guarantee to find services but at the same time the biggest coverage.

In the case of UDDI the public repositories of Web Services were shut down in the beginning of 2006. Recent studies hence conclude “that for publicly available Web Services the UDDI based approach has failed and been discontinued” [37]. Currently Web Services can most easily be found using regular search engines such as Google or specialized engines such as xmethods [62] or seekda [51].

Once a service has been discovered how can a user connect to it? When querying for a service the Web Service Description Language (WSDL) [61] description of the service is returned. Using this description a developer can construct a Simple Object Access Protocol (SOAP) client that connects to the service. SOAP is a platform and vendor independent, XML-based protocol that is used to access services, objects, and servers. A SOAP (XML) message contains an envelope identifying the message as an XML message, header (optional) providing encryption information and a message body. The message body contains all information the message recipient needs – usually method calls and response information.

10.1 Integration through Web Services

10.1.1 Integration with trucking companies

Because of this built in automation, setting up a connection using Web Services is much simpler than using traditional EDI. While EDI requires IT teams and takes months to implement, a connection through Web Services can be set up by very few developers. It is even possible to access a given Web Service simply through a browser window. So if for example a trucking company gets an order to pick up a container the dispatcher could enter the containers number into a field on a terminals website. The dispatcher then will receive information whether the container is ready for pick up. Such a service can easily be implemented using Web Services. For a trucking company, however, that does 1000 such collections a day this is not convenient. Checking the terminals website for the status of each container would be very cumbersome.

With a Web Service implementation the dispatching system itself could access a Web Service at the terminal and check the status of all required containers. So the dispatcher will not need to sit in front of a web browser all day long and there is no chance he may forget to check the status of a particular container – increasing productivity.

Web Services also provide the capabilities to integrate a near ports distribution centers warehouse and yard management with a terminals computer system. The distribution center may be able to achieve a just-in-time delivery of its containers, optimizing and reducing required yard and warehouse space. If the center uses an outside trucking company it also may be integrated using Web Services. The warehouse management system / yard management system could order trucks to bring full containers and remove empty ones. As a result of this integration the warehouse employees will only need to work with their management system – the rest will happen automatically.

10.1.2 Integration with rail facilities

A similar scenario applies to a near dock rail facility. The rail facility could use its own management system to enable the loading and discharging of containers to and from a train and integrate it with the terminal and one or more trucking companies.

10.1.3 Integration with container depots

Also empty containers that are kept in a depot outside the terminal must regularly be brought back to the terminal and loaded onto ships. Integration of the depots computer system with a terminals system and a trucking companies system would allow for example the terminals management system to manage the empty container inventory and help depots by pre-advising them of arriving or needed containers.

10.1.4 Integration with shipping lines

Shipping lines are the primary customers of a terminal. Terminals are essentially agents of the shipping lines that interact with the shipping lines customers on behalf of them. As such they must be aware of the shipping lines business rules and follow them. One example of this would be the rules that apply when a container must be stored on terminal grounds. Some lines allow their customers to store their containers at the terminal for a certain number of days. This number may be different from line to line. The per-day cost is also tiered usually growing over time. In other cases terminals may give a large consignee a fixed number of free TEU's and not care about individual containers. In the end there are many possible scenarios all expressed in business rules that the terminal must be aware of.

The shipping lines will document all the rules and provide the terminals with all the data needed to evaluate the rules for each possible situation. This leads to a never-ending stream of EDI messages. The terminal then must implement these rules in their computer system. This system is expensive to set up, operate and maintain. Also the flexibility of the shipping lines is limited since they must take into account whether the terminals will be able to implement a new business rule fast enough.

Web Service will provide a cheaper and more efficient alternative: A shipping line could simply specify which data it needs to have access to so that it can apply its business rule. When an event occurs at a terminal the terminals computer system would automatically provide the required information to the shipping lines computer system using Web Services. The shipping lines system then would evaluate the data using its business rules and return a response such as for example "Ok to receive" or "denied". The terminal could then apply its own business rules and continue processing the event.

Such an approach would have several immediate advantages:

- The terminal would not need to know the shipping lines business rules.
- The shipping line could change them "on the fly" anytime it wants.
- The terminals costs would be reduced (no need to set up the lines rules).
- The cost of EDI communication would be reduced.

- The lines could ensure that all their rules are enforced in the same way at all terminals they use.

To collect a storage charge the terminal simply hands the shipping line the container number plus some related data and the shipping lines system would return the amount owed or directly take care of billing and payment. This would reduce processing times because now storage charges will be based on the business rules of the shipping line and not on what the terminal believes the rules to be. In the end this could allow a shipping line to modify its storage charges on a bill of landing basis giving it utmost flexibility. Clearly this approach would lower the costs of both terminals and shipping lines while at the same time greatly enhancing their flexibility – a win-win situation.

10.1.5 Integration with Port Communication Systems

Currently at the Los Angeles and Long Beach ports the bilateral information model prevails. Web Services as discussed above are ideally suited to enable communication in this model. But also in the Centralized information Model Web Services can become the backbone of all electronic communication and interaction at the ports. In the Centralized Information Model agents such as terminals, trucking companies, rail companies, government agencies communicate through a Port Communication System (PCS). The PCS provides a central access point at allows all agents to share information such as container availability, trucking company insurance status, truck driver credentials, schedules, news etc. the classical approach to build a PCS is to store all information in a central database. Since the information constantly changes it must be updated all the time. This is usually done via manual entry or through EDI. The data entered is in almost all cases simply a copy of something that is already stored in one of the agent's computer system.

This means that because EDI is used and due to batch processing, data entry errors and security reasons the data is almost never real-time, never exactly the same and rarely comprehensive. Most agents will not want to copy all data they own into the central database. If Web Services are used instead of EDI to communicate with the central database information can be timely and new functions can be added as need be and with little effort. In this architecture when two agents want to exchange data the PCS acts as an intermediary. Both agents send and receive the data via Web Services. Security rules could easily be enforced increasing the willingness for participation in the PCS.

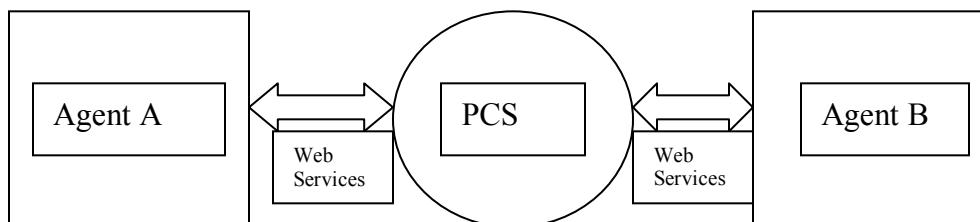


Figure 17: Web Services as portals to PCS

The actual processing happens at the endpoints while the PCS acts as a storage point and information relay.

What are the advantages of tighter integration through efficient electronic communication at the ports?

- Reduced costs – minimize errors, no or little EDI and data entry costs.
- Increased productivity through real-time communication capabilities with rail and trucking companies.
- Increased customer confidence in the terminal through direct integration.
- Better relationship with customers and customers.
- In the long run terminals may be able to increase revenue.

10.1.6 Internal Integration

Finally Web Service can be used to implement communication between internal processes within for example a terminals area of control. They can be used for example to interact and exchange data with a terminal gate or to receive equipment updates.

10.2 Implementation of a Web Service connection

There are many options available when deciding to implement a Web Service connection. Web Service products can be divided into three groups: components, middleware and turnkey.

- **Components** – Components help a developer create a web service. There are many commercially available components (Microsoft, IBM, Sun, Borland and others). There are also open source, free components such as Apache Axis 2[1]. The cost of these components is low (\$0 to \$3000) per developer but they require a major development effort to turn the components into a functioning Web Service. Apache Axis 2 not only supports the SOAP style of Web Services but also the Representational State Transfer (REST) [22] style of Web Services. REST has recently become very popular. With REST domain specific data can be transferred through HTTP without any additional messaging layer such as SOAP or session tracking via HTTP cookies. So an application can interact with a resource by knowing two things: the identifier of the resource, and the action required—it does not need to know whether there are caches, proxies, gateways, firewalls, tunnels, or anything else between it and the server actually holding the information. REST, however is not a new standard – it is simply a style that promotes to implement Web Services in the same way that made the web successful.
- **Middleware** – With middleware legacy systems can be adapted to Web Services. The approach is targeted for larger companies that know how their legacy systems work and that have the resources / IT departments to learn the middleware. The costs is much higher than the cost of components (\$20,000 - \$100,000+). Since these are all inclusive solutions buyers get locked into one particular solution. On the other hand the risks are also much lower compared to components where in house developers carry most of the responsibility.

- **Turn Key Systems** - ERP vendors such as People Soft, Navis or SAP offer turnkey web services support for their applications. These systems are targeted for what the corresponding ERP system does. Compared to middleware or component solutions they offer little flexibility. On the other hand adopters need only very little technical knowledge. A turnkey solution is a practical alternative if a customer can determine in advance that the system exactly fulfills his/her needs. There is a little time needed between acquisition and deployment.

Navis WebAccess [43] is an example of a turn key solution. WebAccess provides information 24-hours a day. Reports are current the moment they are requested, and data is consistent across all parties. Each time a container is moved, a train departs, or information flows through the system, data is instantly updated and available via WebAccess, either on-line, in an e-mail notification or via cellular phone.

WebAccess can provide access to [43]:

- Ship schedules and berth information
- Load and discharge lists
- Container details, status and history
- Chassis inventory
- Equipment availability
- Delivery requirements
- Demurrage information and payments
- Holds (Line, Customs, Agriculture, Off-hire, Service)
- Vessel and barge manifests
- Booking and B/L details
- Truck transactions
- Routing audits
- Appointments
- Interchange agreements
- Damage reports

10.2.1 Consuming a Web Service

Systems that host Web services are called *provider* systems. When a Web service is created it is deployed on a provider system as a service definition. The developer configures the service definition by creating service endpoints. The endpoints hold policies and settings, which enable the *consumer* applications to communicate with it and consume the service definitions. The service definition and the runtime configuration (service endpoints) reside on the same provider system.

A user, a system that wants to consume Web services running on a provider system, must have configured consumer applications on a consumer system. The consumer applications then invoke the functionality provided by the Web services. On the consumer side a developer creates a consumer application (also called consumer proxy), that he deploys on a consumer system. To be able to consume Web services, a business administrator has to configure the consumer proxies at runtime, providing specific settings which are stored into logical ports. The logical port points to the provider system. Moreover, it points to the service endpoint of the configured service definition.

To consume allow applications to consume Web Services directly requires some

programming experience. If a Web Service is accessed infrequently it can be offered through a Web browser and a user may enter data manually and then receive back through the browser window the information he/she requested. This approach could be used for example in case of a small trucking company that needs to pick up a container. If the company however needs to pick up 100 or more containers a day the browser based approach may become cumbersome and inefficient. As a result customers of a terminal may be unwilling to adopt this new technology and in anticipation the terminal may not even initially want to invest into Web Service technologies.

Setting up a direct system-to-system connection so that an application can consume a Web Service provided by another application can be done relatively easily by an experienced developer. In Microsoft's popular .Net framework [40], for example, setting up a direct connection to consume an offered Web Service is a straightforward four step process:

1. Discover and gather information about the service – this can be done by a developer through a web page. The web page (by the service) will hold all the information the developer needs to know to build the proxy class that locally will represent the offered service.
2. Generate a proxy class of the service – this is a simple local place holder for the remote services offered by the Web Service.
3. Use the created proxy class to invoke an available service – in this step the developer sets up the connection between the local placeholder and the actual service.
4. Write an interface for the service – finally the developer makes the proxy available locally so that it can be invoked automatically whenever needed.

Other frameworks have similar approaches. In all frameworks the developer on the consumer side does not need to know any details about the implementation of the Service on the provider side. This *decouples* the consumer from the provider – allowing both to experience the benefits of information sharing while at the same time preserving their independence.

10.3 Web Service Security

A provider must protect its Web Services and only provide them to a client with proper authentication credentials. To achieve this, SOAP (which is based on XML) is used to send authentication information with Web Service commands. In the SOAP header username and password information are passed along so that only the users a provider chooses can access the service.

Digital signatures can also be used to sign documents. A consumer can sign a document or request with his/her private key and send it along with the payload of the message. The provider is then able to verify the signature with the consumer's public key to validate the authenticity of the document or request. One key benefit of signing is the concept of non-repudiation. In addition the provider is able to keep a copy of the signature. With it the provider will later be able to prove that this document/request was really signed by this consumer. The XML Signature standard provides a means for signing parts of XML documents, providing end-to-end data integrity across multiple systems.

One key benefit of signing is the concept of non-repudiation. When transactions are performed, it is often necessary to be able to prove that a particular action took place. Using signatures, service providers can not only provide evidence that a document is valid but also record the message transactions into signed audit logs. Once an audit log has been signed it cannot be modified without significantly changing the signature. Hackers often modify audit logs in order to "cover their tracks" to avoid detection. Signed log files can be used to detect such cases.

When third party non-repudiation is required, digital receipts provide independent verification that specific transactions have occurred.

Since Web Services are accessed and provided over the Internet encryption is often necessary to protect shared information and requested services. Standard SSL encryption using HTTPS allows point-to-point data privacy between Web service consumers and service providers. However, in many cases, the service provider may not be the ultimate destination for a request. A service provider itself may act as a service requestor, requesting information from other service providers. This means that even though the request is encrypted, the provider must be able to recognize that it is not the ultimate destination of the request without having to break secrecy by decrypting the request. This is possible since the XML Encryption standard permits encryption of portions of the message allowing header information to be used for routing purposes while leaving the sensitive payload encrypted. Sensitive information can then be left encrypted to the ultimate destination, allowing true end-to-end data privacy.

11. Recommendations and Conclusions

In this report we discussed (1) XML/EDI, (2) EDIINT (Web EDI), (3) The language M and (4) the use of Web Services as potential options for XML based communications at the ports. We briefly summarize the earlier discussed main costs / benefits of these systems.

(1) XML/EDI: This is by far the cheapest solution. In XML/EDI business documents are exchanged using email, ftp or http/https. This allows the exchange over the Internet eliminating the need for Value Added Networks. Overall, however, besides moving away from VAN's there are no other clear advantages. Namely the main problem – and reason for EDI's inefficiency – EDI's lack of exception handling support remains with this approach. So it is more of a patchwork solution. The same can be said about the in-house development of EDI-XML conversion tools. Collaboration EDI: Collaboration EDI does not simply exchange information but connects business processes with each other. It standardizes the business process and is not just a simple message standard. EbXML [19] is an attempt to internationally standardize collaborative XML/EDI. Its success so far has been at best mixed. While initially there has been a lot of enthusiasm about ebXML, adoption has been slow. The main reason for this can likely be found in the fact that ebXML does not allow the execution of Web Service Business Process Execution language (BEPL). Processes in BEPL export and import information by using Web Service interfaces exclusively. This is a major drawback since more and more companies are moving towards offering some services as Web Services. So a company would have to invest into an expensive standard (ebXML) only to discover that it does not cover the complete spectrum of Service Oriented Architectures (SOA). It appears that since 2006 there is little movement on the ebXML front.

(2) EDIINT: This approach is effective but requires the purchase of a system that can support EDIINT. The system has real time capabilities but requires a conversion to the appropriate XML format. So in essence the problem of many different EDI standards is simply shifted to many different XML formats with the effect that as in the case of EDI exceptions (errors, changes in procedure etc.) must be handled manually. So in essence it has similar properties and problems like a classical EDI system.

(3) The language M: M is based on XML and aims to address the interoperability issues of XML. It was created at the MIT Data Center [10] as an open, global language that communicates between proprietary schemas enabling companies to combine, visualize and understand data. Like a regular spoken language. M has a dictionary to describe the meaning of words. The dictionary consists of a collection of definitions that can be used when making computer transactions. The dictionary of M also includes word relations, data format, and language translations. These all help to form and understand messages written in M.

While M appears to solve many of the problems that XML has, it was just recently released as a prototype and is not yet ready for commercial applications. Moreover M is useful as a medium that makes different XML schemas such as XHTML[63], SOAP[9], SMIL[53], ebXML[19] and RosettaNet[46] obsolete. If one of these schemes, however, becomes the dominating one, M loses its significance. At this moment it appears that SOAP and Web Services could become this dominating scheme.

(4) Web Services: This is clearly the most promising approach.

Based on personal communication, many terminals at the Los Angeles and Long Beach ports are currently in the process of or have recently installed expensive and comprehensive Terminal Operating Systems (TOS) such as Navis Sparcs N4. These TOS represent large investments and hence in any realistic analysis must be considered as a baseline from where to build further connectivity.

It appears almost certain that terminals at the Los Angeles and Long Beach Ports will continue to favor the bilateral information model. There is very little likelihood that terminals anytime soon will want to convert to the centralized information model. This means that any direct conversion from EDI to XML would have to occur in a bilateral fashion, on a case by case basis. Even though – based on our experiments – conversion tools could be developed in house by a few skilled developers, we believe that most – if not all – terminals would view this as a very risky step and will hence not pursue it. Over the last few decades terminals have gotten used to the continued investments into EDI communication they need to make. In general they write it off as one of these costs that cannot be avoided. Also in the case of terminals the use of EDI is limited. It is mostly used in communication with shipping lines / carriers and rail companies. EDI allows these agents to quickly transmit manifests and communicate business rules. Even though the terminals are all aware that EDI is not an ideal solution – it requires a continued investment in manpower – they do not seriously question these costs since all their competitors face the same costs. As a result they do not feel at a disadvantage even though EDI is far from efficient and cost effective.

This thinking, however, is bound to change. The continued investment in state-of-the-art Terminal Operating Systems such as Navis Sparcs N4 is evidence of this fact. Once one terminal operator moves to such a TOS that allows it to easily integrate all events on the terminal grounds into its centralized computer system, its competitors will begin to view this as a serious business advantage and will have no other choice than to move into the

same direction. This process has already begun at the Los Angeles and Long Beach ports. The APM terminal is as of February 2009 in the process of automating its operations using Navis Sparcs. The current economic downturn may slow down this process but it can also provide an opportunity to experiment with these techniques while business is slower to be ready for future growth.

These new TOS provide support for both EDI and Web Services. Internally the APM terminal, for example, already uses XML / Web Services to communicate the status of terminal equipment to the TOS. If terminals become aware of the tremendous advantages of Web Services in particular, their thinking may change and some terminals may begin to move from EDI to Web Services to communicate with carriers / shipping lines and rail operators.

Since terminals have already invested into large and powerful TOS we believe that there is no reason to develop an in house EDI to XML conversion. Such an effort may simply lead to a shift of the problem from EDI to XML since pure XML also has serious interoperability issues. Instead we recommend to seriously explore the additional connectivity that TOS's provide. In particular we believe that Web Services (which are based on XML communication) are extremely promising as foundational building blocks for port communication. They do not have XML's interoperability issues since any consumer can get information about a services interface in advance and does not need to be concerned about the services implementation. Hence they will allow the terminals to become true brokers of information that provide all the services that carriers, rail operators, truckers, shipping line customers and government agencies require. From the viewpoint of a terminal there seems to be no reason not to assume this role. A Web Service enabled terminal would be able to operate far more efficiently than a terminal that uses only EDI. All concerned parties will be able to access and provide information in real time. But for this conversion to really take place, the other agents such as carriers / shipping lines and rail operators will also have to offer Web Services so that a TOS can find and request automatically the information it requires to do its job. Once this happens there will be no more need to use EDI. Until then EDI – besides its well-documented deficiencies – will continue to persist at the ports.

	Cost	Benefit
EDI	<ul style="list-style-type: none"> • Requires continued human intervention • Expensive since cannot handle exceptions automatically • Expensive since allows only bilateral communication • Not real-time 	<ul style="list-style-type: none"> • Small file size • Well established
Web Services	<ul style="list-style-type: none"> • Must be set up either in house (moderate), through middleware (more expensive) or 	<ul style="list-style-type: none"> • Allows for automatic system to system communication

	<p>using turn key system (most expensive)</p> <ul style="list-style-type: none"> • All agents must participate for it to be effective • May require expensive state of the art Terminal Operating Systems. 	<ul style="list-style-type: none"> • No or little human intervention needed • Same information can be made accessible using the same tool to many participants • Participants share only what they want to share • No need for a service consumer to know the implementation of a service • Service provider can change implementation on the fly and does not need to communicate change to consumers (e.g. a carrier can change its business rules) • Based on XML so can also use browser to view and access it • Does not have XML's interoperability issues since interface is easily learnable
--	--	---

Table 5: Cost/benefits of EDI / Web Services

12. Implementation

We recommend implementing the research findings by making Web Services the basis of all electronic port communications. Because of the large investments that terminal operators, shipping lines and rail companies have made in EDI it is unrealistic to expect that they will immediately move away from EDI. We therefore suggest two implementation phases:

Phase 1: In this phase we recommend that the terminals provide connectivity to truckers, trucking companies, government agencies, shipping line customers (shippers), carriers and rail companies through Web Services using web browsers. All parties will then be able to retrieve information about a container, a pickup, a shipment through a Web Browser, that is

be able to receive real-time status information directly from the terminal. Using a state of the art TOS such as Navis Sparcs N4 this can be set up relatively easily since N4 is Web Services enabled. Other comparable state of the art TOS have similar properties or allow users to purchase a module that enables Web Services. We believe that this will provide a great initial improvement of port connectivity.

All interested and authorized parties will be able to retrieve real-time information enabling them to base decision on current conditions.

Today many cell phones or PDA's are internet enabled so a truck driver, for example, will be able to retrieve updated information about a container he is supposed to pick up even on his way to a terminal or at the terminal while waiting. Confidentiality of data transmission can be ensured by making the data only accessible through https.

The implementation of this phase can commence immediately.

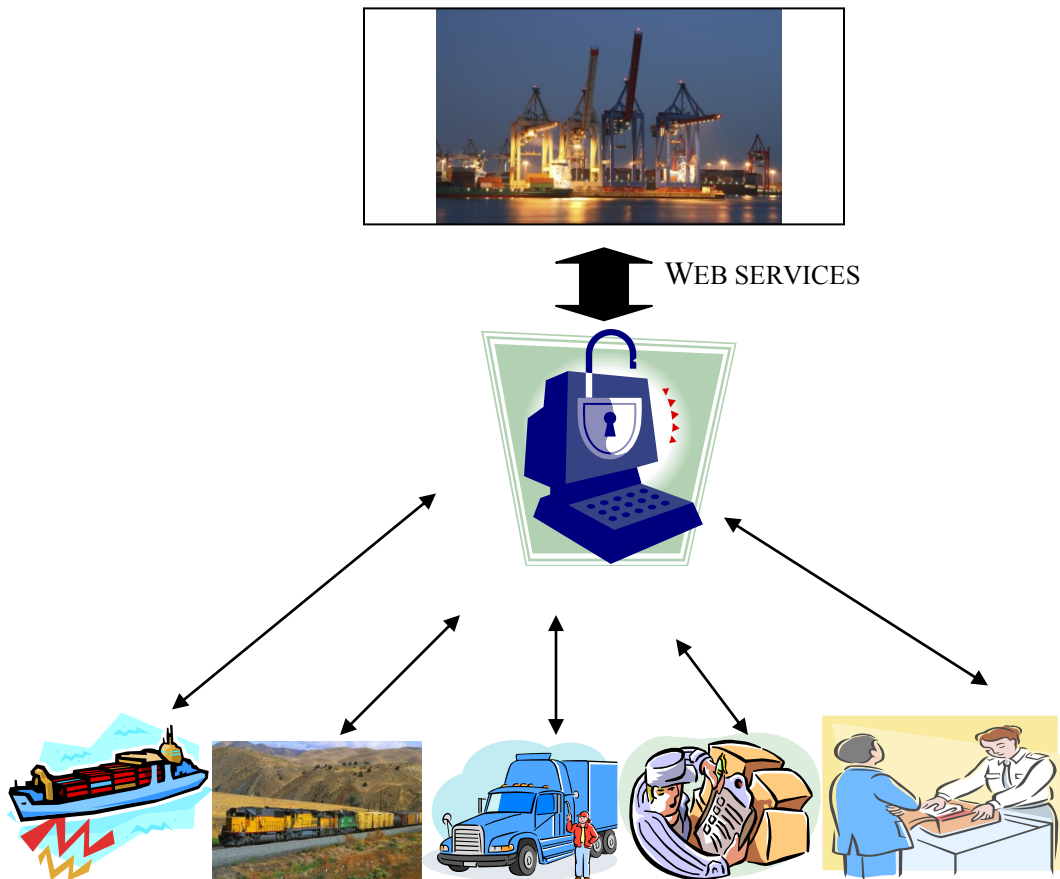


Figure 18: Web Services enabled access to a Terminal Operating System (Phase 1)

Phase 2: In phase two we recommend that trucking companies, government agencies, shipping lines/carriers and large shippers modify their internal Operating Systems or Business Management Systems so that they can also become Web Service providers. This will require some investments but each company will have a choice how much it wants to invest. A company can either use in house developers to make the modifications, buy a middleware tool that when plugged in provides the Web Service connectivity or buy a turnkey solution. Which solution a company chooses depends on its business model and the advantages it sees in allowing direct access (computer to computer) to its own computer system.

For a small trucking company this may be a low priority but if it needs to keep track of several hundred container pick ups and drop offs a day the benefits may quickly outweigh the costs. For a shipping line on the other hand the benefits are much clearer. Allowing a terminals OS to connect to the shipping lines computer system directly through Web Services would allow the shipping line to change its business rules without having to inform the terminal directly. The shipping line would simply change the implementation of its Web Services and the terminal would get the updated information whenever it consumes the service. We believe that such an approach could greatly simplify the interactions between shipping lines and terminals. Since now terminals are always aware of the latest business rules they can act as true information brokers that provide information on a shipping lines behalf to the shipping lines customers. With Web Services information can flow unobstructed between computer systems virtually eliminating transmission errors, allowing for real-time information flow, and significantly reducing human involvement. With thousands of containers being moved every year we expect a significant return on investment. Namely containers will be able to flow faster and more efficiently through the supply chain cutting down costs and increasing capacity.

We anticipate that the implementation of this phase will begin once the benefits of the implementations of phase one have become evident to all participants.

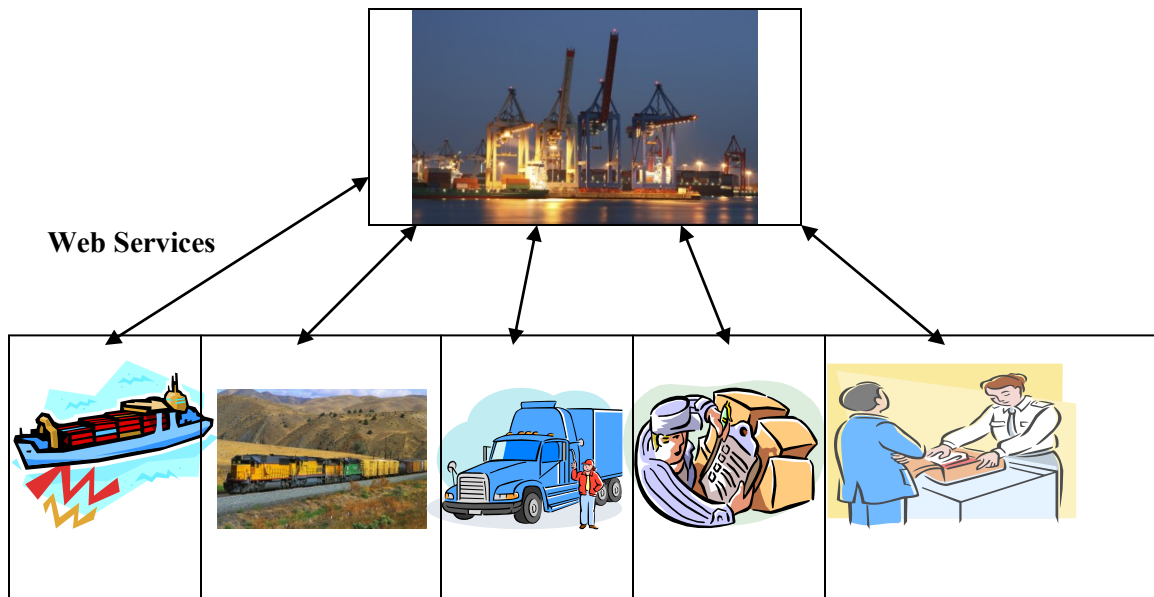


Figure 19: Web Services enabled computer to computer communication (Phase 2)

13. References

- [1] Apache Axis 2 available at: <http://ws.apache.org/axis2/>
- [2] S. Banerjee, D. Golhar, Electronic Data Interchange: Characteristics of users and non users, Information and Management 26(1) 65-74 (1994).
- [3] J. Berje. The EDIFACT Standards. NCC Blackwell, 1991.

- [4] BizTalk Mapper:
<http://msdn2.microsoft.com/enus/library/ms865624.aspx#bts:serializer>
- [5] BizTalk Server. Available at [http:// www.microsoft.com/biztalk/default.msp](http://www.microsoft.com/biztalk/default.msp)
- [6] BMEcat. E-Business Standard, available at <http://www.bmecat.org>
- [7] Bolero trade communication platform. Available at <http://www.bolero.net>
- [8] BOTS Open Source EDI translator, available at
<http://bots.sourceforge.net/en/index.shtml>
- [9] A. Bosworth, D. Box, M. Gudgin, M. Nottingham, D. Orchard and J. Schlimmer.
XML, SOAP and binary data. Available at:
<http://www.xml.com/pub/a/2003/02/26/binaryxml.html>
- [10] D.L. Brock, E.W. Schuster and T.J. Kutz Sr., An Overview of the M language, MIT-DATACENTER-WH-009, January 2006, (<http://datacenter.mit.edu/MIT-DATACENTER-WH-009.pdf>)
- [11] bTrade TDAccess/TDPeer available at <http://www.btrade.com>
- [12] Canoo Ultralight Client, available at www.canoo.com
- [13] Commerce XML. Available at <http://www.cxml.org>
- [14] COPRAR message, available at <http://www.smdg.org>
- [15] D. Crocker. Mime Encapsulation of EDI Objects, TC RFC1767.
<http://www.rfceditor.org> (1995).
- [16] e-Business W@tch, The European E-business Report 2004 Edition: A Portrait of Ebusiness in 10 Sectors of the EU Economy (<http://www.ebusiness-watch.org>).
- [17] e-Business W@tch, The European E-business Report 2005 Edition: A Portrait of Ebusiness in 10 Sectors of the EU Economy (<http://www.ebusiness-watch.org>).
- [18] EDIINT White paper. Available at http://www.effective-data.com/edi_white_papers.htm
- [19] Electronic Business XML (ebXML) available at <http://www.ebxml.org>
- [20] M. Emmelhainz, Electronic Data Interchange: does it change the purchasing process? Journal of Purchasing and Materials Management 23(4) 2-8 (1987).
- [21] J. Farrell, G. Saloner, Standardization, compatibility, and innovation, RAND Journal of Economics 16 (1) (1985) 70-83.

- [22] R. Fielding, Architectural Styles and the Design of Network-based Software Architectures, Doctoral Dissertation, University of California Irvine, 2000.
- [23] D. Foxvog and C. Bussler. Ontologizing EDI: First Steps and Initial Experience. Proceedings International Workshop on Data Engineering Issues in E-Commerce (DEEC 2005), 2005.
- [24] D. Foxvog and C. Bussler. Ontologizing EDI Semantics: First Steps and Initial Experience. Proceedings International Workshop on Ontologizing Industrial Standards (OIS 2006), 2006.
- [25] GEntran Integration Suite available at <http://www.sterlingcommerce.com/Products/Integration/Gentran+Integration+Suite/>
- [26] S. Gosain, A. Malhotra, O.A. El Sawy and F. Chehade, The impact of common e-business interfaces, Communications of the ACM 46 (12) (2003), pp. 186–195.
- [27] Groovy, available at: <http://groovy.codehaus.org/>
- [28] T. Harding and R. Scott. FTP Transport for Secure Peer-to-Peer Business Data Interchange over the Internet. <http://www.ietf.org>, 2006.
- [29] C.T. Hsieh and B. Lin, Impact of standardization on EDI in B2B development, Industrial Management & Data Systems 104 (1) (2004), pp. 68–77.
- [30] C. Iacovou, I. Benbasat, A. Dexter, Electronic Data Interchange and small organizations: Adoption and impact of technology, MIS Quarterly 19(4), 465-485 (1995).
- [31] Inovis BizConnect available at <http://www.inovis.com/solutions/software/bizmanager/bizconnect/>
- [32] International Trade Trends and Impacts: Los Angeles County Economic Development Corporation: The Southern California Region: <http://www.laedc.org/reports/Forecast-2009-02.pdf>
- [33] ISIS European XML/EDI pilot project available at <http://www.tieke.fi/isis-xmledi/>
- [34] ISO. Open-EDI Reference Manual. ISO/IED JTD 1/SC30 ISO Standard 14662, 1997.
- [35] K. Kanakamedala, J. King, G. Ramsdell, The truth about XML, McKinsey Quarterly (3) (2003) 9-12.
- [36] A. Kotok, D. Webber, ebXML the new global standard for doing business over the Internet”, New Riders, September 2001.
- [37] Holger Lausen and Thomas Haselwanter. Finding Web Services. 1st European Semantic Technology Conference, 2007.

- [38] F. Lehmann. Machine-negotiated Ontology-based EDI. Electronic Commerce: Current Research Issues and Applications, 1996.
- [39] m-e-c eagle EDI converter, available at: <http://mec-eagle.sourceforge.net/>
- [40] Microsoft .Net framework, available at: <http://www.microsoft.com/NET/>
- [41] T. Mukhopadhyay, S. Kekre, s. Kalathur, Business value of information technology: a study of electronic data interchange, MIS Quarterly 19(2), 137-156 (1995).
- [42] Navis Sparcs TOS, available at www.navis.com
- [43] Navis WebAccess, available at <http://www.navis.com/webaccess.jsp>
- [44] J-M. Nurmilaakso, EDI, XML and e-business frameworks: A survey, Computers in Industry 59, 370-379 (2008).
- [45] J-M. Nurmilaakso, J. Kettunen, I. Seilonen, XML-based supply chain integration. Integrated Manufacturing Systems 13/8, pp. 586-595, 2002.
- [46] Partner Interface Processes (RosettaNet) available at <http://rosettanet.org>
- [47] D. Power, Supply chain management integration and implementation: a literature review, Supply Chain Management: An International Journal 10 (4) (2005), pp. 252–263.
- [48] B. Prasad, Role of XML in transportation:
http://www.wipro.com/datadocs/whitepaper/XML_in_Transportation.pdf
- [49] G. Premkumar, K. Ramamurthy, S. Nilakanta, Implementation of electronic data interchange: an innovation diffusion perspective, Journal of Mangment information Systems 11(2) 157-186 (1994).
- [50] K. Reimers, Standardizing the new e-business platform: learning from EDI experience, Electronic Markets 11 (4) (2001), pp. 231–237.
- [51] <http://seekda.com/browse>
- [52] S. Shim, V. Pendyala, M. Sundaram and J. Gao. Business-to-business e-commerce frameworks. IEEE Computer, Vol. 33 No. 10.
- [53] SMIL, Synchronized Multimedia Integration Language, available at <http://www.w3.org/TR/REC-smil/>
- [54] K. Srinivasan, S. Kekre, T. Mukhopadhyay, Impact of electronic data interchange technology on JIT shipments, Management Science 40(10) 1291-1304 (1994).
- [55] Erik Tiemroth, Building a competitive advantage with Web Services, available at <http://www.navis.com>

- [56] TranXML XML standard for shippers. Available at <http://www.transxml.org>
- [57] UDDI version 3.0, available at <http://uddi.org>
- [58] Virtuele Haven – A blueprint for a virtual port (Port of Rotterdam)
<http://www.virtuelehaven.nl>
- [59] Web Service, available at: <http://www.w3.org/2002/ws/>
- [60] T. Wilson, EDI is alive and kicking, study says, InternetWeek (February 21) (2000).
- [61] WSDL, available at: <http://www.w3.org/TR/wsdl>, March 2001.
- [62] <http://www.xmethods.net>
- [63] XHTML 1.0: The Extensible HyperText Markup Language. Available at: www.w3.org/TR/xhtml1/
- [64] XML, Extensible Markup Language, available at www.w3.org/XML
- [65] XML/EDI Introduction Guidebook, Electronic Commerce Promotion Council of Japan, 2003.