HELSINKI UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering

Juha-Miikka Nurmilaakso

XML-based Supply Chain Integration: A Review and a Case Study

A thesis to fulfill the requirements for the degree of Licentiate of Science in Technology.

Supervisor: Professor (pro tem) Timo Soininen

HELSINKI UNIVERSITY OF TECHNOLOGY	ABSTRACT OF THE LICENTIATE THESIS

Author and name of the thesis:
Juha-Miikka Nurmilaakso
XML-based Supply Chain Integration: A Review and a Case Study

Date: 30.11.2003 **Number of pages:** 56 + 6

Department: Computer Science and Engineering
Professorship: T-86 Information Technology

Supervisor: Professor (pro tem) Timo Soininen

The thesis studies the extent to which Extensible Markup Language (XML) supports supply chain integration Both a conceptual research with a literature study and a constructive research with a case study were used. Supply chain integration is about information sharing between trading partners. It is an important part of efficient supply chain management. The point-to-point, hub-and-spoke and service-oriented models are the basic integration models of supply chains. The basic XML technologies enable straightforward exchange of data in a supply chain. XML is useful in syntactic interpretation but insufficient in semantic interpretation. Therefore, e-business frameworks are necessary in supply chain integration. An e-business framework answers at least one of the questions of what information should be shared, when, and how. The e-business framework deals with business documents, business processes, or messaging needed in information sharing between the trading partners. 26 XML-based e-business frameworks were studied, eight of them were compared, and three were described in detail. Categories of document-centric, cross-industry, industry-specific, and process-centric frameworks were recognized.

The thesis also presents an XML-based integration system prototype that was implemented and tested at ABB Control and InCap Electronics in Finland. In addition to business documents, XML was used for describing “technical” processes. Since ABB and InCap used EDIFACT standard for exchanging purchase orders, purchase order responses, and invoices, the prototype were evaluated against Electronic Data Interchange (EDI). The corresponding XML documents were based on the XML Common Business Library (xCBL) framework, and the XML documents for purchase order lists and demand forecasts were developed. Since XML enables customized business documents, and the prototype uses the Internet instead of the value added network, the prototype was more flexible to implement and operate than EDI. In addition, the engine-processor architecture together with its XML-based configuration facilitated the maintenance of the prototype. Business benefits of the prototype were highly case-specific but its use provided significant cost savings in comparison to EDI. The prototype had lower implementation costs than EDI. On the other hand, a small or medium-sized enterprise does not necessary need an integration system of its own but it can use the large enterprise’s XML-based integration system by a browser. Operating costs were also lower for the prototype than for EDI. Comparing XML and EDI shows that there are more XML-based e-business frameworks than EDI standards. In this sense, XML is more flexible than EDI. However, a large number of the frameworks causes difficulties. There are indications that XML-based integration is less expensive than EDI-based integration. Unfortunately, the comparison between XML and EDI may be biased, and the experiences from XML-based supply chain integration are quite limited in general. Although XML does not remove all the integration problems, XML-based supply chain integration can be a significant alternative to EDI. XML does not guarantee but it can promote a shared understanding of business documents and business processes.

Keywords:
Supply chain, Integration, XML, E-business framework, Prototype

Table of contents

List of publications

This work is based to large extent on the following four publications, which are referred to in the text by their roman numerals:

I Jansson, K., Karvonen, I., Mattila, V. P., Nurmilaakso, J., Ollus, M., Salkari, I., Ali-Yrkkö, J., Ylä-Anttila, P., 2001, *Uuden tietotekniikan vaikutukset liiketoimintaan (Impacts of Modern Information Technology on Business)*. Teknologiakatsaus 111, National Technology Agency of Finland, Helsinki, Finland. Pp. 28-32.

II Seilonen, I., Nurmilaakso, J. M., Jakobsson, S., Kettunen, J., Kuhakoski, K., 2001, "Experiences from the development of an XML/XSLT-based integration server for a virtual enterprise type co-operation". In Thoben, K. D., Weber, F., Pawar, K. S. (Eds.), *Proceedings of the 7th International Conference on Concurrent Enterprising: Engineering the Knowledge Economy Through Co-operation*. ICE 2001, June 27-29, Bremen, Germany. Centre for Concurrent Enterprising, Nottingham, United Kingdom. Pp. 321-328.

III Nurmilaakso, J. M., Kettunen, J., Lehtonen, J. M., Saranen, J., Seilonen, I., 2001, "Seamless production planning and communication in distributed manufacturing: Case ABB switchgear production". In Stanford-Smith, B., Chiozza, E. (Eds.), *E-Work and E-Commerce: Novel Solutions and Practices for a Global Networked Economy*, Vol. 2. e-2001, October 17-19, Venice, Italy. IOS Press, Amsterdam, the Netherlands. Pp. 867-873.

IV Nurmilaakso, J. M., Kettunen, J., Seilonen, I., 2002, "XML-based supply chain integration: A case study", *Integrated Manufacturing Systems*, Vol. 13, No. 8 (Special Issue on Enabling Supply Chain Integration Using Internet Technologies), pp. 586-595.

Abbreviations

ASC X12	Accredited Standards Committee X12
API	Application Programming Interface
B2B	Business-to-Business
B2C	Business-to-Consumer
B2G	Business-to-Government
BNF	Backus-Naur Form
BOV	Business Operational View
BPML	Business Process Modeling Language
BPSS	Business Process Specification Schema
CA	Communication Application
CRM	Customer Relationship Management
cXML	Commerce XML
DOM	Document Object Model
DTD	Document Type Definition
EAI	Enterprise Application Integration
ebCPP	ebXML Collaboration Partner Profile
ebMS	ebXML Messaging Services
ebRIM	ebXML Registry Information Model
ebRS	ebXML Registry Services
ebXML	Electronic Business XML
EDI	Electronic Data Interchange
EDIFACT	Electronic Document Interchange for Administration, Commerce, and Transportation
ERP	Enterprise Resource Planning
FSV	Functional Service View
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SLL
HTTP/S	HTTP or HTTPS
IFV	Implementation Framework View
MIME	Multipurpose Internet Mail Extensions
PIP	Partner Interface Process
OAGIS	Open Applications Group Integration Specification
ODBC	Open Database Connectivity

OWL	Web Ontology Language
PDM	Product Data Management
RDF	Resource Description Framework
RDFS	RDF Schema
RNBD	RosettaNet Business Dictionary
RNIF	RosettaNet Implementation Framework
RNTD	RosettaNet Technical Dictionary
S/MIME	Security Multiparts for MIME
SAX	Simple API for XML
SCM	Supply Chain Management
SGML	Standard Generalized Markup Language
SME	Small and Medium Sized Enterprise
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SSL	Security Socket Layer
UDDI	Universal Description, Discovery, and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VAN	Value Added Network
WSDL	Web Services Definition Language
xCBL	XML Common Business Library
XML	Extensible Markup Language
XMLDSIG	XML Digital Signatures
XPDL	XML Process Description Language
XSDL	XML Schema Definition Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformation

1 Introduction

1.1 Background

The move towards e-business has an impact on organizations in every industry. Supply chain integration affects the way companies do business with their customers, suppliers and any other partners. Many companies are implementing business-to-business (B2B) initiatives aimed at automating trading relationships and making them more efficient. There is a variety of such initiatives from simple exchange of transactional information to complex supply networks (Jones 1997, Koski et al. 2001). The benefits of automated B2B interactions have been recognized for a long time. Compared to mail, phone, and fax, automated B2B interactions save money and time. They are faster and less error prone than manual interactions. In addition, automated B2B interactions release human labor resources from monotonous work. Of course, taking into account the implementation and operating costs of automated B2B interactions, they do not always reduce the overall costs but sometimes they only redistribute them.

Currently, the most widely used B2B communication method is Electronic Data Interchange (EDI), which was established over 25 years ago (Copeland and Hwang 1997). EDI is commonly defined as the direct computer-to-computer exchange of business documents, such as purchase orders and invoices, with messages in a predetermined format. These messages are produced from the data in the sender’s information system, transmitted electronically, and automatically entered into the receiver’s information system. The key idea of EDI is that the B2B communication takes place without human intervention.

EDI provides advantages compared to manual B2B communication methods. It has shown that electronic data exchange between the trading partners leads to faster transactions and cost savings (Pawar and Driva 2000). In addition, cross-industry standards American National Standards Institute's Accredited Standards Committee X12 (ASC X12) (ASC 2003) in North America and United Nations Electronic Document Interchange for Administration, Commerce, and Transportation (EDIFACT) (UNECE 2003) in the rest of the world, as well as multi-party communication connections in Value Added Networks (VANs) have promoted the use of EDI (Copeland and Hwang 1997).

Despite its widespread adoption, EDI is costly and rigid (Goldfarb and Prescod 2002). Its expense and inflexibility has limited its spread to the large enterprises and acted as an effective barrier to implementations by small and medium sized enterprises (SMEs). Many SMEs use EDI only because their larger partners require it. EDI has had the additional disadvantage that it does not support real-time communication but business documents are exchanged using batch processing. It is not unusual that a company uses EDI to communicate with a small fraction of its trading partners, whereas communication with the remaining partners relies on other methods (Westarp et al. 1999). In addition, many company use EDI to exchange purchase orders and invoices but other methods to exchange other business documents.

The Internet offers a cheap and flexible approach to B2B interactions. In principle, it provides a channel for establishing relationships with a broad range of trading partners.

This could enable companies to build on the progress they have already made using EDI in integrating their supply chains. The Internet could open up the opportunities for these benefits to be enjoyed by SMEs that have not used EDI. For example, the following statistics shows that the use of the Internet has spread in a different way compared to the use of EDI.

A survey of e-business in Nordic companies conducted in 2000 (Statistics Denmark et al. 2001) shows that the volume of EDI sales of all Finnish companies was 23.0 billion Euros, whereas the volume of Internet sales was 1.8 billion Euros. Although the volume of EDI sales accounts for more than tenfold the volume of Internet sales, five per cent of all Finnish companies made at least one per cent of their total turnover from orders received via EDI, whereas 10 per cent made it from orders received via homepages. When questions about the relevance of barriers regarding Internet sales were asked, 48 per cent of Finnish companies regarded considerations for existing sales channels, and 45 per cent products not suitable for selling via homepages as an important barrier. This survey also shows that the use of EDI greatly depends on the enterprise size and has concentrated on the largest enterprises. SMEs have been more eager to adopt the Internet than EDI. Respectively, the use of the Internet to sell products is related much to the industry. The difference between the volumes of EDI and Internet sales indicates the growth potential of the use of the Internet in B2B interactions.

In principle, it should be easy for companies to start communicating electronically with their trading partners. In practice, it can be far from straightforward to implement even relatively simple B2B interactions with a small number of partners because any two companies differ in their communication infrastructure, in the way they describe their goods and services, and so on. The Internet alone does not solve this integration problem. Tim Bray, a co-author of the Extensible Markup Language (XML) standard, has claimed, "XML is the ASCII of the future". Although Bray may exaggerate the significance of XML, many are convinced that XML will be the ASCII standard of e-business (Economist 2001). According to Goldfarb and Prescod (2002), XML will enable more efficient B2B communication than EDI. Tens of e-business frameworks have been standardized to tackle the integration problems (e.g. Hasselbring and Weigand 2001, Shim et al. 2000). An e-business framework answers at least one of the questions of what information should be shared, when, and how. XML-based e-business frameworks utilize XML in business documents, business processes, or messaging.

A large number of companies have directed their attention to integration systems, which utilize both the Internet and XML. These systems are application servers (e.g. BEA 2003, IBM 2003, Microsoft 2003, SoftwareAG 2003, Tibco 2003, webMethods 2003) that provide solutions for integration problems within and between companies. This integration takes place through middleware instead of back-end integration with “legacy” systems, like enterprise resource planning (ERP), supply chain management (SCM), customer relationship management (CRM), and product data management (PDM). There are high hopes that such integration systems would enable supply chain integration in cases in which it has been impossible or unprofitable. However, it is not clear that XML-based supply chain integration will satisfy these hopes.

Since the mid-1990s, Internet technologies have created increasing technical opportunities for B2B communication between different companies. The integration systems represent an applied type of Internet technologies that is becoming popular.

Java and XML form a basic type of these technologies that is already popular. On the other hand, companies in many industries have changed their operations into forms of co-operation that could benefit from the increased communication opportunities. These two parallel developments provide the motivation for studying a match between them.

1.2 Goals

The thesis aims to test a hypothesis that the XML-based integration systems provide a sound basis for supply chain integration. In other words, XML-based integration would be more flexible and less expensive than EDI-based integration. The thesis also strives to correct possible misunderstandings of XML-based supply chain integration.

- The primary goal of the thesis is to examine XML-based integration systems and how they support supply chain integration between companies. These experiences are expected to be useful for both the developers and users of the XML-based integration systems.
- The secondary goal of the thesis is to give a review of supply chain integration, basic XML technologies, and e-business frameworks. The thesis analyses e-business frameworks, expectations, and experiences.

1.3 Methodology

The research approaches applied in the thesis can be classified by the goals of the thesis. A constructive approach and a case study are applied to the primary goal, a conceptual approach, and a literature study to the secondary goal. The thesis takes a computer science and engineering viewpoint rather than an industrial engineering and management viewpoint.

A constructive approach has a long tradition in the engineering research and is widely used in computer science. It provides a natural basis for the primary objective because the purpose of this objective is to produce a novel solution to a practically and theoretically relevant problem. In addition, a case study was needed to validate this solution. Although it cannot be statistically generalized, it is strong on realism.

The constructive approach was employed by implementing a prototype of the XML-based integration system in a research project named Virtual Switchgear Factory, which was a subproject of the GNOSIS Virtual Factory project. This prototype called Communication Application (CA) was studied with an industrial case, in which the main contractor of the production network was a switchgear production company, ABB Control. The other companies of the network were subcontractors and suppliers of the main contractor. One of these subcontractors, InCap Electronics, was also involved in the case. The CA was developed for and experimented with ABB and InCap. Since these companies partly used EDIFACT in supply chain integration, it was possible to evaluate the benefits and costs of the XML-based integration system. Research scientists not involved in the implementation of the CA carried out this evaluation for the main part. This was sensible for two reasons. On the one hand, developers of the CA had no sufficient expertise of conducting such an evaluation. On the other hand, it was necessary to minimize the potential bias resulting from expertise on the CA. Therefore, the developers kept in the background in the evaluation process.

The evaluation was conducted in the form of a two-tier cost-benefit analysis. The first phase of the evaluation was a case study, whereas the second phase focused on the generalization of the case-specific results. The idea was to produce more generic information on the potential benefits and costs of the integration system.

The evaluation work was carried out in close cooperation with ABB Control. This provided an invaluable opportunity to test new ideas and solutions in practice and to get immediate feedback on their feasibility with regard the needs of ABB and InCap Electronics. Since ABB and InCap were using EDI for exchanging a part of their purchase order, purchase order response, and invoice data over the VANs, the case also enabled a comparison between CA and EDI with regard to their support for operations and costs.

The evaluation methodology was largely developed as the work in the case progressed. Important methodological references included Anandarajan and Wen (1999), Luoma et al. (1999), Willcocks and Lester (1999a, 1999b), and Zuboff (1988). The main methodological challenges related to the identification and assessment of the potential benefits and costs of a system that was not operational at the time of the evaluation. In consequence, special emphasis was given to developing methods that would make it easier to concretize what a corresponding operative system would be able to do if it was actually made into a product and brought into production use at ABB. Therefore, not only demonstrations and experiments with the CA but also discussions and workshops with the representatives of the case companies had an important role in the evaluation process. This resulted in a rich picture of the case companies' processes and systems, and the potential role of a CA type of integration system in that domain. Different kinds of models, such as ABB's order-delivery process models and interaction diagrams to describe functionality of the CA, were used to facilitate discussions and analysis.

For the most part, the evaluation results were based on consensus. For example, the results of functional tests were jointly investigated because the work required a profound understanding of EDIFACT and XML Common Business Library (xCBL) 2.0 (xCBL 2003), which was used as an XML-based e-business framework in the case. Estimates on the expected implementation demands and potential business impacts were mainly founded on the views of the case companies. Configuration and maintenance related demands and costs were estimated by the evaluator, who installed and maintained the CA at ABB, and the developers of the CA based on their own experience. Estimates on the EDI-related benefits and costs were provided by ABB's system specialists with extensive EDI development experience.

There were no particular reasons to worry about possible respondent bias in this case because neither the representatives of the case companies were responsible for implementing the prototype, nor were they committed to bringing it into production use. It was concluded that the given estimations were sincere and based on the best available knowledge and experience at that time. In addition, representatives of the case companies as well as developers of the CA reviewed the evaluation results. Some of these results were based on confidential information and, therefore, were not published in the original form.

For part of the secondary goal, a conceptual approach is applied to supply chain integration issues to illustrate different kinds of integration models. A literature study reviews e-business frameworks based on their documentation. This review includes analysis to identify the basic features of the e-business frameworks, a comparison and classification of these frameworks.

1.4 Scope

For the primary goal, the thesis is limited for two reasons. Originally, there was no intention to carry out detailed evaluation of the CA in the supply chain context because only ABB Control was officially involved in the Virtual Switchgear Factory project as an industrial partner. InCap Electronics was kindly willing to participate in the experiments and evaluations. During these experiments, the CA resided at ABB and ABB's and InCap's users had access over the Internet to the CA by a browser. Interactions between the two integration systems were not tested in this industrial case because it was not possible to install and maintain the CA at InCap. In all, alternative options were few because only ABB and InCap were able to provide the information needed.

A lack of research scientists capable of implementing or evaluating the CA resulted in another major restriction with the project budget and schedule. Therefore, a number of important security and reliability features were not implemented or evaluated in the CA. These features include the use of Security Socket Layer (SSL) in data communication and transaction management in database operations. In addition, the industrial case focused on only six interactions although a large number of important interactions were identified. Interactions related to purchase orders, purchase order responses, and invoices were chosen. These are the most common EDI messages in Finland (Kärkkäinen et al. 2001). xCBL 2.0 was chosen because it was ahead in the development of XML-based e-business frameworks.

For the secondary goal, the thesis covers those XML technologies used to validate, parse, and transform XML documents. This thesis does not deal with Web Services standards Simple Object Access Protocol (SOAP) (W3C 2000b), Universal Description, Discovery, and Integration (UDDI) (UDDI 2002), and Web Services Definition Language (WSDL) (W3C 2001a) or Semantic Web standards Resource Description Framework (RDF) (W3C 1999c), RDF Schema (RDFS) (W3C 2003a), and Web Ontology Language (OWL) (W3C 2003b) although their importance seems to be increasing. The family of the XML standards is large but many of them are still in the unstable state. At least, tools with any real use are missing. This instability also applies to e-business frameworks. These frameworks usually contain a lot of documentation. This thesis gives a detailed view on xCBL, RosettaNet (2003), and Electronic Business XML (ebXML) (ebXML 2003) because they have been more in the limelight. These frameworks are compared to Business Process Modeling Language (BPML) (BPMI 2003), Commerce XML (cXML) (cXML 2003), Open Applications Group Integration Specification (OAGIS) (OAG 2003), papiNet (2003), and XML Process Description Language (XPDL) (WfMC 2003), which also seem to be suitable for industrial B2B e-commerce.

1.5 Structure

Section 2 focuses on the secondary goal of the thesis. Subsection 2.1 deals with supply chain management (I), and subsection 2.2 with XML technologies (IV). The other subsections are new contributions that summarizes expectations and experiences concerning XML-based supply chain integration, and analyses, compares and classifies XML-based e-business frameworks.

Section 3 addresses the primary goal. It reports experiences gained from the implementation and evaluation of the prototype in an industrial case (II, III, IV).

Section 4 compares this research with previous research, discusses its results, presents opinions about XML-based supply chain integration, and presents questions for further research.

Section 5 concludes the thesis.

2 Review

This section gives a review of XML-based supply chain integration. Supply chain integration is a necessary rather than a sufficient requirement for efficient supply chain management. XML provides a medium to facilitate information sharing between the trading partners. XML is not a complete solution for information sharing but e-business frameworks are needed to fill the gaps in interoperability. First, the section discusses the concept of supply chain management, presents three reasons for and three impacts of supply chain management, and identifies three integration models. Next, this section goes through the XML standard and the basic XML technologies for validating, parsing, and transforming XML documents. In addition, the section explains why e-business frameworks are necessary, and identifies three basic features, and four categories of the frameworks. This section summarizes 26 XML-based e-business frameworks, compares the eight most important frameworks, and describes three of them in detail. In addition, the section looks over expectations on and experiences from XML-based supply chain integration. Finally, XML technologies, expectations, e-business frameworks, and experiences are reviewed critically.

2.1 Supply chain management

2.1.1 Concept

A *supply chain* means a flow of goods, services, money, and information through different units (Tan 2001). These units are legally independent companies, physically distant factories or offices, or organizational entities having autonomy to make decisions over their information systems. *Supply chain management* encompasses logistics that studies material and information flows, purchasing, and selling in terms of operative questions, such as transportation, ordering and packing, as well as strategic questions, such as competition. Although there are a large number of definitions for supply chain management, it is a more comprehensive concept than logistics. Supply chain management plans and controls various flows from the raw material suppliers to the end customers.

Literally speaking, the concept of supply chain management is problematic because supply refers to operations management through push, i.e. inventories and supplier control. In comparison, demand chain management reflects operations management through pull, i.e. orders and customer control. Operations management is not independent of the general economic situation. During recessions, supply-based management is often utilized because of overcapacity and more intense competition. On the other hand, undercapacity and less intense competition lead to demand-based management during booms. The concept of supply chain management is associated both with supply-based and demand-based management.

Supply chain management is closely related to Porter's (1985) idea of a value chain that is based on the process view of organizations. According to this idea, an organization can be seen as a system that is made up of subsystems, each with inputs, transformation processes and outputs. The Supply Chain Operations Reference (SCOR) model (Supply

Chain Council 2003) is a description of processes in the supply chain. It consists of the following processes:

- *Plan*: Demand/supply planning
- *Source*: Sourcing/material acquisition, manage sourcing infrastructure
- *Make*: Production execution, manage make infrastructure
- *Delivery*: Order management, warehouse management, transportation and installation management, manage deliver infrastructure

This model requires that participants in the supply chain share their information. These participants are not only different units within the same company but they often belong to the different companies. The former case is related to an *internal supply chain* because one participant has authority over other participants. The latter case is related to an *external supply chain* because all participants have autonomy. The latter case is in many ways more difficult than the former one. Since a supply chain is based much on cooperation, competition is not necessary between companies but between supply chains. However, supply chains are not isolated but they often cross and form a *supply network*. Figure 1 illustrates that a company may have a number of units and may be involved in a number of supply chains at the same time. The company cooperates with its customers and suppliers that may compete with each other. Similarly, these customers and suppliers may cooperate with the competitors of the company. There is a tradeoff between competition and cooperation. Since management is about decision making, which requires information, supply chain management requires integration, which is about information sharing between the participants. Therefore, the basic challenges originate from the difficulties in balancing competition and cooperation between the participants in the supply chains.

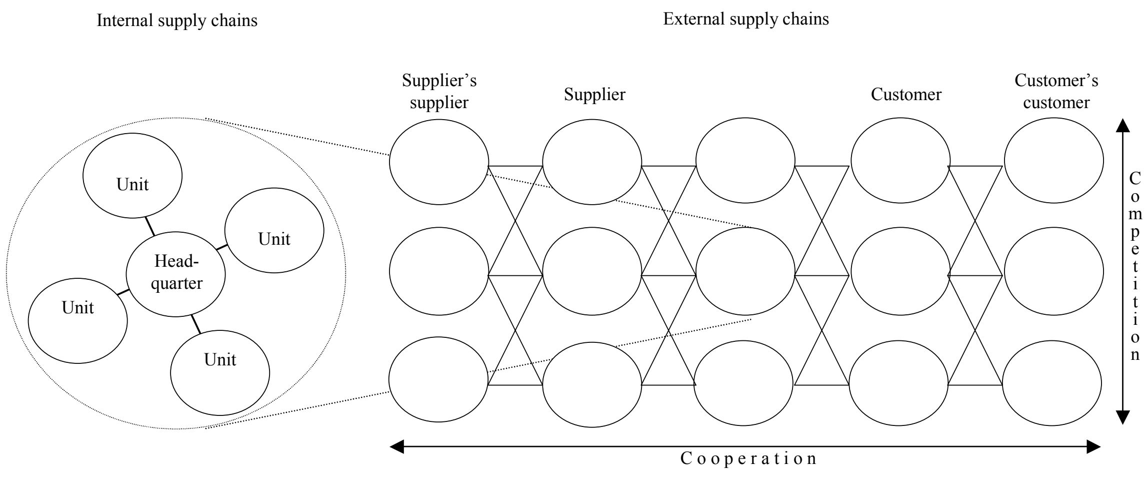

Figure 1: Competition and cooperation in a supply network

2.1.2 Business impacts

Supply chain management aims to intensify the processes from raw-materials suppliers to end customers. Its purpose is to increase the added value and to improve the resource utilization and cost efficiency by getting the right product at the right time to the right place with a minimum handling and buffering. The following reasons emphasize the role of supply chain management:

- *Internationalization of companies*: Globalization has lead to a situation, in which companies, even medium-sized ones, have units in different countries. Since operations often take place over a geographically wide area, needs arise to economize warehousing and transportations.
- *Complex products*: Products have become complex and a single company does not have the necessary resources to realize them. Since development of resources from scratch may take a lot of time and efforts, it may be the best choice to utilize the existing resources of other companies. Although a product is not complex, it often has to be customized.
- *Changes in market conditions*: Rapid technological development has resulted in shortening product life cycles. Since profitable opportunities come and go quickly, the market is saturated fast and new products have to be brought onto the market frequently. Therefore, the needs and abilities of different trading partners have to be found and matched quickly.

Supply chain management has impacts on the following objectives at least:

- *Response time* is the time when the customer places an order and receives this order. Since shorter response time increases customer satisfaction, it improves competitiveness.
- *Inventory level* has an important effect on profitability because a smaller inventory reduces the working capital costs. It also reduces the risk of obsoleteness.
- *Lead time* is the sum of the processing time to convert raw materials into finished products and the waiting time at the buffers. In principle, competitiveness and profitability should not be mutually exclusive. In practice, increasing safety inventories may reduce the response time. Similarly, a longer response time may enable smaller inventories. Since a shorter lead time enables faster reactions, it opens up a way to improve both competitiveness and profitability.
- *Capacity utilization* plays an important role. On the one hand, underutilization erodes both competitiveness and profitability because idle employees and machines generate costs but no revenue. It is possible to balance these costs to the revenues but adjustment of capacity may also be costly. On the other hand, overutilization wears out capacity fast. Although it would be profitable in the short term, deterioration of capacity may be very costly in the long run. In order to improve capacity utilization, the company has to be capable of selling its own capacity when that is possible and buying capacity from others when it is needed.

Supply chain management performs accurate assignment of resources and exact synchronization of activities in the supply chain. It has potential for considerable positive or negative business impacts. In the best case, supply chain management enables a seamless supply chain that reduces the response time, lead time, and inventory level, and improves capacity utilization. In the worst case, its failure leads to redistribution of rewards and risks that is costly and creates no new value added.

2.1.3 Integration models

As information sharing, integration can be categorized in many different ways. The first basic classification deals with the nature of communication.

- *Manual communication* means human-to-human communication between the trading partners. Human intervention is needed because this communication takes place by meetings, mails, phone calls, faxes, and e-mails. The fact is that technology can never replace all social aspects related to this kind of communication.
- *Semi-automated communication* is human-to-system communication. Exchange of information is performed in one end by the information system, and in the other end, human intervention is necessary.
- *Full-automated communication* means system-to-system communication. Information is exchanged between the trading partners' systems automatically. No human intervention is needed.

The second basic classification is based on the nature of the trading partners.

- *B2B e-commerce* occurs between organizational units of the company or between companies.
- *Business-to-consumer (B2C) e-commerce* takes place between companies and households.
- *Business-to-government (B2G) e-commerce* occurs between companies and public institutions.

Integration can be internal or external. Internal integration occurs within an organizational unit. It is integration of applications from different software vendors and in-house systems with packaged applications. *Enterprise application integration* (EAI) focuses on these issues. Linthicum (2001) suggests five approaches to internal integration. These are data-oriented, application interface-oriented, method-oriented, portal-oriented, and process integration-oriented solutions.

External integration takes place between organizational units. These units are different units within the company or different companies. External integration includes sharing information with customers and suppliers as well as activities brokered through intermediaries. *Supply chain integration* is related to these issues. Goldfarb and Prescod (2002) suggest four approaches to external integration. In *traditional commerce*, each customer and supplier may be automated internally. These customers and suppliers connect their systems by manual processes, such as mail, fax, and phone calls. Through a *web storefront*, the customers can view a supplier's catalog of goods and services and place orders directly into a supplier's system. Nothing is necessary automated on the customer's side. With an *e-commerce portal*, customers go to the portal website to view the supplier's catalogs and place orders. Suppliers also go to the same website to view and respond to orders. In *integrated e-commerce*, the systems of different companies exchange information directly, which eliminates manual processes.

Without loss of generality, three models can be presented for supply chain integration. The first two model represent the prevailing solutions, whereas the third model reflects some recent developments in the B2B e-commerce. The following models are independent of whether it is a question of bilateral or multilateral relationships.

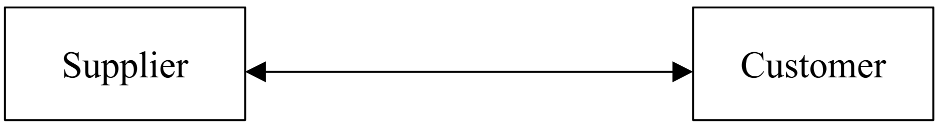

Figure 2: The point-to-point model

Figure 2 presents the *point-to-point model*. This model is traditional because EDI solutions rely on it. According to the point-to-point model, integration takes place directly between the trading partners who have signed a trading partner agreement, which defines and describes B2B interactions. There may be a number of brokers in between but their role is to route the messages. The point-to-point model enables flexibility to automate different kinds of interactions. Standards are not a necessity but they reduce the implementation costs. If a point-to-point solution is trading-partner-specific, the implementation costs may be very high. In that case, a profitable solution requires a long-term, high volume trading relationship.

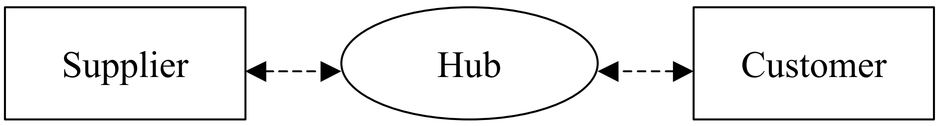

Figure 3: The hub-and-spoke model

Figure 3 illustrates the *hub-and-spoke model* that was born along with the Internet. Usually this model is based on an electronic marketplace. There are three kinds of electronic marketplaces. Buyer-side marketplaces are more likely in the B2B and B2G contexts and seller-side marketplaces in the B2C context. Third-party marketplaces are not limited to one of these contexts. For example, FastParts (www.fastparts.com) for electronic components, MetalSite (www.metalsite.com) for metals, PaperSpace (www.paperspace.com) for pulp and paper, Commerce One Net (www.commerceone.net), and Ariba Supplier Network (service.ariba.com) are third-party B2B marketplaces, whereas General Motors' TradeXchange for automobile components is a buyer-side B2B marketplace. Project portals also reflect the hub-and-spoke model in the broad sense. According to the hub-and-spoke model, the trading partners are integrated through a hub. The hub is not only a broker but also an intermediary aggregating a demand for the suppliers and a supply for the customers. The trading partner agreement is prepared between the intermediary and the trading partner manually. The hub-and-spoke model is limited to interactions, such as product searches, catalog updates, orders, and auctions in the electronic marketplaces and read and write operations of the electronic notice board in the project portals. In order to reach a critical mass of trading partners, standards are necessary.

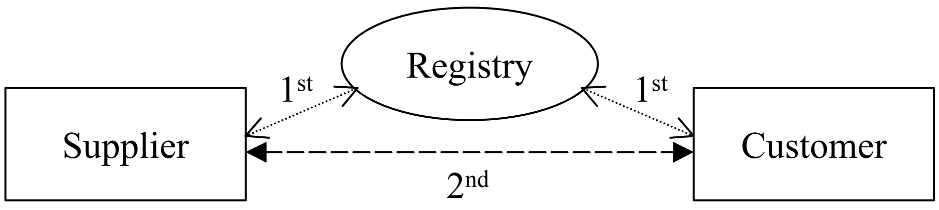

Figure 4: The service-oriented model

Figure 4 presents the *service-oriented model* that has received much publicity along with XML. With this respect, ebXML (2003) and Web Services standards SOAP (W3C 2000b), UDDI (UDDI 2002), and WSDL (W3C 2001a) have been important. Trading partners, i.e. a customer, a supplier, or an intermediary, are integrated by a party who possesses a registry. This registry contains information about the business and technical capabilities of the trading partners. In the first phase, the trading partners store their capabilities into the registry and retrieve an intersection of capabilities supported by both of these trading partners. A trading partner agreement is generated automatically. In the second phase, these partners perform interactions according to this intersection. The service-oriented model requires standards. Their expressive power determines what kind of interactions can be performed. The service-oriented model may resemble a combination of the point-to-point and hub-and-spoke models. However, the registry is a kind of an electronic directory. Interactions do not pass through the registry but are executed directly between the trading partners.

2.2 XML technologies

2.2.1 XML standard

The XML standard (W3C 2000a) was designed to improve the functionality of the Internet by providing flexible information structuring. XML standard 1.0 was introduced in 1998. XML is extensible because it is not a fixed format like Hypertext Markup Language (HTML) (W3C 1999b) but a metalanguage for describing other languages. XML can be utilized to design customized markup languages for different types of documents. XML is a subset of Standard Generalized Markup Language (SGML) (ISO 1986), with some exceptions. SGML is a standard for defining descriptions of the structure of an electronic document. SGML is very powerful but complex, whereas XML is a lightweight version of SGML cleansed of all the features that make SGML too complex for the Internet. SGML is very comprehensive, which makes it hard to learn and expensive to implement.

The XML standard defines the syntax of a markup language that is applied to represent the structure of an electronic document. These documents are formed of a set of objects that contain elements. Each element may have a number of attributes, according to which the document will be processed. XML provides a formal syntax to describe the dependencies between the objects, elements and attributes, and to build an electronic document. Namespaces were added to XML shortly after the XML Standard 1.0. A namespace is a collection of names for a particular domain with specific meaning. The same name may be used in several namespaces with different meanings because uniqueness of the namespaces is achieved using Uniform Resource Identifiers (URIs).

Figure 5a shows an example XML document without namespaces, and 5b with a namespace. PurchaseOrder is a root element of this document, other elements are non-empty, and Unit is an attribute of the element Quantity. This document is well formed. Why do namespaces matters? In Figure 5b, a namespace http://www.soberit.hit.gi/b2b may define that the element ProductID means a product name as a string, whereas there may exist other namespaces in which this same element means a product code as a number.

```
<?xml version="1.0" encoding="utf-8"?>
<PurchaseOrder>
  <BuyerParty>
    <PartyID>X</PartyID>
  </BuyerParty>
  <SellerParty>
    <PartyID>Y</PartyID>
  </SellerParty>
  <Product>
    <ProductID>ZZZ</ProductID>
    <Quantity Unit="kgs">
      12.3
    </Quantity>
  </Product>
</PurchaseOrder>
```

(a)

```
<?xml version="1.0" encoding="utf-8"?>
<PurchaseOrder xmlns="http://www.soberit.hut.fi/b2b">
  <BuyerParty>
    <PartyID>X</PartyID>
  </BuyerParty>
  <SellerParty>
    <PartyID>Y</PartyID>
  </SellerParty>
  <Product>
    <ProductID>ZZZ</ProductID>
    <Quantity Unit="kgs">
      12.3
    </Quantity>
  </Product>
</PurchaseOrder>
```

(b)

Figure 5: Example of an XML document

The XML document has to be well formed. The XML document has only one root element that is the parent element of other elements in the document. A non-empty element must have a start-tag and an end-tag in between the start-tag and end-tag of its parent element. It has either child elements or content in between its start-tag and end-tag. An empty element has an empty-tag but neither child elements nor content. A tag always starts with a left angle bracket (<), contains the element name, and ends with a right angle bracket (>). In fact, an end-tag begins with a left angle bracket and slash (</) and an empty-tag ends with a slash and right angle bracket (/>). XML is also case-sensitive. Only start-tags and empty-tags can contain attributes. If there is an attribute, the attribute name must be followed by an equal sign (=) and an attribute value enclosed in single (') or double quotes ("). XML is a context-free language that can be represented in the Backus-Naur form (BNF) (Appendix 1).

In practice, the XML standard alone is not enough but a number of XML technologies are necessary in supply chain integration. Figure 6 illustrates one possible case in supply chain integration, in which business documents are exchanged between trading partner X's and Y's databases. The idea is to employ basic XML technologies for validating, parsing, and transforming XML documents as business documents.

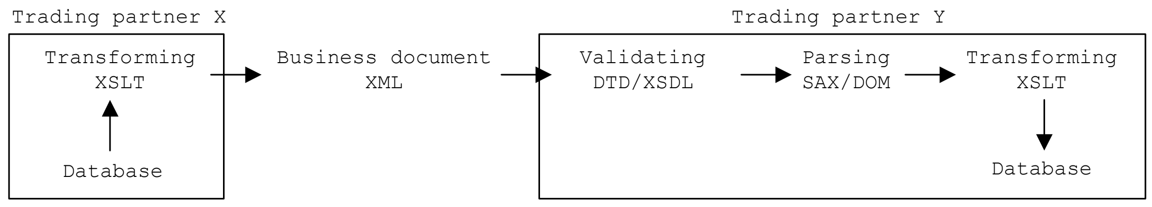

Figure 6: Basic XML technologies in supply chain integration.

2.2.2 Validating: DTD and XSDL

An XML document can be validated against a Document Type Definition (DTD) or schema that is included in or referenced by the document. Since DTD and schemas describe the metadata of the document, they can be used to define a vocabulary that is a shared specification for documents in a particular domain of interest.

Although DTDs are a part of XML Standard 1.0, they originate from SGML. A DTD specifies the structure of the XML document by defining elements of the document, one, zero-or-one, zero-or-more, and one-or-more occurrences of the elements and the hierarchical order between the elements. The DTD may define required and optional attributes of the elements and alternative values of the attributes. It may also contain references to other DTDs. Unfortunately, DTDs are not well-formed XML documents and provide little support for data typing, cardinality, and namespaces. A schema is an XML document for describing the structure of XML documents. XML Schema Definition Language (XSDL) (W3C 2001b), which is also known as XML Schema, is an XML language for schemas. XSDL offers a number of built-in datatypes and capabilities of defining datatypes. It allows to apply datatypes to both element content and attribute values. In addition to XSDL, there are also other schema languages, such as Microsoft's XML Data-Reduced, Schematron, and RelaxNG, which are used less frequently. Figure 7a shows an example DTD and 7b an example XSDL.

```
<!ELEMENT PurchaseOrder (
  BuyerParty,
  SellerParty,
  Product+)>
<!ELEMENT BuyerParty (
  PartyID)>
<!ELEMENT SellerParty (
  PartyID)>
<!ELEMENT Product (
  ProductID,
  Quantity)>
<!ELEMENT PartyID
  (#PCDATA)>
<!ELEMENT ProductID
  (#PCDATA)>
<!ELEMENT Quantity
  (#PCDATA)>
<!ATTLIST Quantity
  Unit (kgs|lbs)
  #REQUIRED>
```

(a)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="PartyID" type="xs:string"/>
  <xs:element name="ProductID" type="xs:string"/>
  <xs:element name="Quantity">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:decimal">
          <xs:attribute name="Unit" type="QuantityUnit"
             use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="PurchaseOrder">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="BuyerParty"/>
        <xs:element ref="SellerParty"/>
        <xs:element ref="Product"maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
<xs:element name="BuyerParty">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="PartyID"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SellerParty">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="PartyID"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Product">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ProductID"/>
        <xs:element ref="Quantity"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="QuantityUnit">
    <xs:restriction base="xs:string">
      <xs:enumeration value="kgs"/>
      <xs:enumeration value="lbs"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

(b)

Figure 7: Example of DTD and XSDL

Although not all XML parsers are validating, the most popular ones enable that XML documents are validated against DTDs. In comparison, a number of XML parsers supporting validation against XSDL is small but increasing.

2.2.3 Parsing: SAX and DOM

There are two approaches for parsing XML documents. Simple API for XML (SAX) (SAX 2002) is an event-based application programming interface (API) that reports parsing events, such as the start and end tags, directly to the application through callbacks. The application implements handlers to deal with the different events. Since the original SAX did not support namespaces, SAX2 was developed. Document Object Model (DOM) (W3C 2002) is a tree-based API that converts an XML document into a tree structure. The application has access to navigate and manipulate this structure. It can also generate a well-formed XML document.

Comparing the parsing approaches, the SAX requires more programming due to handlers and makes it harder to visualize XML documents than the DOM. However, the SAX is faster and less memory-intensive because it does not load entire XML documents as tree structures into the memory.

There are several XML parsers for parsing XML documents. The most popular XML parsers support both SAX and DOM approaches.

2.2.4 Transforming: XSLT

XSL Transformation (XSLT) (W3C 1999a) is an XML language for transforming XML documents into other XML documents. XSLT is not intended as a complete general-purpose XML transformation language but it is designed for use as a part of Extensible Stylesheet Language (XSL), which is a stylesheet language for XML. XSL includes a vocabulary for specifying formatting. For example, the block formatting represents the breaking of the content of a paragraph into lines.

A transformation expressed in XSLT describes the rules for transforming a source document into a result document. This stylesheet contains a set of template rules that consist of patterns and templates. This allows a stylesheet to be applicable to a wide class of documents that have structures similar to the source document. A pattern is matched against elements in the source document. A template is instantiated to create the part of the result document that is separate from the source document. In constructing the result, elements from the source can be filtered and reordered, and arbitrary structure can be added. Figure 8a shows an example of an XSLT document and 8b is the output document of the transformation of the document in Figure 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8"
    indent="yes"/>
  <xsl:strip-space elements="*"/>
  <xsl:template match="PurchaseOrder">
    <PurchaseOrder>
      <BuyerPartyID>
        <xsl:value-of select="BuyerParty/PartyID"/>
      </BuyerPartyID>
      <SellerPartyID>
        <xsl:value-of select="SellerParty/PartyID"/>
      </SellerPartyID>
      <ProductID>
        <xsl:value-of select="Product/ProductID"/>
      </ProductID>
      <Amount>
        <xsl:value-of select="Product/Quantity"/>
      </Amount>
    </PurchaseOrder>
  </xsl:template>
</xsl:stylesheet>
```

(a)

```
<?xml version="1.0" encoding="UTF-8"?>
<PurchaseOrder>
  <BuyerPartyID>X</BuyerPartyID>
  <SellerPartyID>Y</SellerPartyID>
  <ProductID>ZZZ</ProductID>
  <Amount>12.3</Amount>
</PurchaseOrder>
```

(b)

Figure 8: Example transformation from one format to another

Previously, an XSLT processor was a separate tool. Currently, many XML parsers are capable of XSLT processing.

2.3 Expectations

The literature provides a number of expectations for the benefits of XML in supply chain integration. The following list reviews the most common arguments for XML.

- *XML is flexible* (Cummins 2002, Fitzgerald 2001, Goldfarb and Prescod 2002, Reimers 2001): Rather than being fixed in describing a particular set of data, XML with its DTDs or schemas is able to define a number of documents that together form a separate language. XML documents can easily be extended by adding further elements and attributes. Documents can be defined by sharing a new DTD or schema within the user community. Since XML is simple in its nature, requirement for considering an XML document well-formed is to follow basic syntax rules.

- *XML is human-readable* (Cummins 2002, Fitzgerald 2001, Reimers 2001): XML documents can be read and written by humans unlike the documents created by applications in a machine-readable format.
- *XML is self-describing* (Cummins 2002, Goldfarb and Prescod 2002): The DTD or schema can guarantee at the time of document creation that all the elements are correctly specified in the right order. The usage of a DTD or schema can help guarantee that the values of the contained elements are valid and fall within acceptable ranges. Documents can be validated at the time of creation or at the time of receipt and be rejected or accepted without human intervention.
- *XML is structured* (Fitzgerald 2001, Goldfarb and Prescod 2002): The nature of XML is a structured document format that represents not only the information to be exchanged but also the metadata encoding the structure of the information to be exchanged. Most text files cannot offer this simple advantage because they represent the information to be exchanged without the metadata. File formats, such as tab-delimited text files, contain data in predefined locations in the file. Relational databases have similar advantages but their formats are only machine-readable.
- *XML is widespread and inexpensive* (Cummins 2002, Fitzgerald 2001): XML tools have become relatively widespread and inexpensive because XML shares many of the properties of SGML and HTML.
- *XML is platform-neutral and widely supported* (Cummins 2002, Fitzgerald 2001): XML is a platform-neutral data format. This means an XML file is the same thing regardless of the platform that processes it. As an open-source technology, XML provides developers a wide base of resources that can provide assistance in the implementation.
- *XML-based systems have lower costs* (Goldfarb and Prescod 2002, Reimers 2001): Compared to EDI, XML offers serious cost and development timesaving. There are many open source tools for document creation, validation, parsing, and transformation.
- *XML separates processing from content* (Goldfarb and Prescod 2002): Since XML represents information and the metadata about the information, it does not specify any particular way for how the data should be processed. In contrast, other formats, such as text files and databases, explicitly require accessing the documents in a specific way. Of course, the data contained within XML needs to be processed but the method for processing this information is not specified anywhere within the XML document.

2.4 E-business frameworks

2.4.1 Basics

Companies draw up their documents and perform their processes in different ways. The differences in meanings of terms and modes of operation between companies result in errors. The companies also have different information systems. Heterogeneous systems cannot exchange information but a lot of manual work is needed to prepare the input for the various systems to process the output. Not only companies are “islands on automation” but units within a company may face the basic problem of supply chain integration, which is information sharing.

To operate across organizational boundaries, trading partners must have shared understanding of their ways of doing business. To automate business interactions, the trading partners’ systems must be capable of communicating. The trading partners have to know what information should be shared, when, and how. Interoperability of business documents, business processes, and messaging is an answer to the questions of information sharing. An *e-business framework* is a standard for this purpose. In a non-philosophical sense, the framework is a kind of ontology that enables communication between systems in a way that is independent of the individual system technologies, architectures, and application domain. The framework often combines other standards, specifications, and classifications.

E-business frameworks cover business and technical aspects of business documents, business processes, and messaging in supply chain integration but they are not limited to these issues. Since only a few of frameworks deal with all business documents, business processes, and messaging issues, frameworks are not only substitutes but in part complements. The following list outlines the basic interoperability issues:

- *Business document issues* are about what information to share. The framework contains a vocabulary that describes the structures and parts of the business documents, and defines meanings of the terms to be used in these documents. For example, if trading partner X sends a purchase order to trading partner Y, this document includes elements for a customer’s name, a supplier’s name, a product name, and an ordered quantity as well as an attribute for a measuring unit.
- *Business process issues* are about when to share information. The frameworks take different approaches to these issues. The *rough process approach* explains in which order to exchange particular business documents. The *detailed process approach* describes the purpose of particular business processes and the trading partners’ roles in them. It also defines what kinds of business documents are needed and in which order to exchange them. The *generic process approach* deals with neither particular business processes nor the exchange of particular business documents. It provides a way to model a combination of the roles, actions, and interactions as choreography of the business process. For example, if Y has received a purchase order from X, Y sends a purchase order response to X.
- *Messaging issues* are about how to share information. This also includes how to handle basic exceptions, such as message losses. Since the message is an envelope consisting of headers, attachments, and content, the framework defines the structure and parts of the headers, and packing, e.g. Multipurpose Internet Mail Extensions (MIME), security, e.g. Security Multiparts for MIME (S/MIME), and transportation standards, e.g. Hypertext Transfer Protocol (HTTP), to be used with these messages. For example, if X and Y exchange

purchase orders and purchase order responses, they use HTTP over SLL (HTTPS) to transport these documents.

2.4.2 XML and frameworks

Since many e-business frameworks utilize XML, this raises the question of the division of labor between XML and the frameworks. Automated document communication comprises the syntactic and semantic interpretations (Russell and Norvig 1995) that are necessary for understanding of the business documents. The syntactic interpretation is divided into lexical and syntactic analyses and the semantic interpretation into semantic and pragmatic analyses:

- *Lexical analysis* scans the characters of the business document and produces tokens according to the lexicon that defines the acceptable combinations of characters in the language. Since XML does not define element and attribute names as well as element contents and attribute values, DTDs or schemas are needed. A framework may provide these DTDs or schemas.
- *Syntactic analysis* obtains the tokens and generates a tree of symbols according to the grammar that defines the structure of the language. If the meanings of the symbols are independent on this structure, XML is enough. Otherwise, the framework needs to provide DTDs or schemas.
- *Semantic analysis* recognizes what the symbols denote and associates the alternative meanings to the symbols in the tree. XML, DTDs, and schemas alone cannot help if the symbol is ambiguous or non-descriptive. The framework defines the meanings for each symbol that are relevant for supply chain management.
- *Pragmatic analysis* interprets the symbols in the prevailing context. For each symbol, the most suitable meaning is chosen from the alternative ones taking into account other symbols and their alternative meanings. Since XML, DTDs, and schemas are context-free languages, their assistance is incomplete in this choice. The framework should guide the choice of the meaning. However, if the knowledge related to this choice cannot be explicated, it is difficult to record this knowledge in the framework.

The following example illustrates these analyses. Trading partner X retrieves a purchase order from the database and sends it. Trading partner Y receives and stores this purchase order in the database.

```
<PurchaseOrder>
  <BuyerPartyID>X</BuyerPartyID>
  <SellerPartyID>Y</SellerPartyID>
  ...
</PurchaseOrder>
```

(a)

```
<PurchaseOrder>
  <PartyID>X</PartyID>
  <PartyID>Y</PartyID>
  ...
</PurchaseOrder>
```

(b)

```
<PurchaseOrder>
  <BuyerParty>
    <PartyID>X</PartyID>
  </BuyerParty>
  <SellerParty>
    <PartyID>Y</PartyID>
  </SellerParty>
  ...
</PurchaseOrder>
```

(c)

Figure 9: Lexically invalid, syntactically invalid, and valid document

In Figure 9, each XML document is well formed but Y understands only business document 9c. This document should be in accordance with the DTD or schema in Figure 7. An e-business framework provides a lexicon in a DTD or schema. In Figure 9a, a lexical analysis does not recognize the elements BuyerPartyID and SellerPartyID. Since these elements are not in the lexicon, the document is lexically invalid. The framework also provides a grammar in a DTD or schema. In Figure 9b, a syntactic analysis finds out that the element PartyID is in the lexicon but its position does not match with the grammar. Since the elements BuyerParty and SellerParty do not exist, the document is syntactically invalid. Figure 9c presents a valid XML document to be stored in the relational database. The framework provides a vocabulary. According to the vocabulary, the element PurchaseOrder means the document is a purchase order. It also tells that BuyerParty contains the customer’s and SellerParty the supplier’s information, and PartyID a trading partner’s name. Using the vocabulary, Y can map the document into a table related to the purchase orders in the database. However, a semantic analysis is not enough to map the data into correct columns in this table. For a pragmatic analysis, the vocabulary may say that if PartyID is within BuyerParty, its content is a customer’s name, and if this element is within SellerParty, its content is a supplier’s name. Since Y is a supplier, it is important to ensure that the purchase order was intended to Y before its data is recorded by a row having X in a column related to the customer name in the table. This example shows that more information may be necessary to deploy applications for supply chain integration that an XML document is able to convey.

In addition to business documents, there are business processes and messaging. XML does not have a self-evident role in business process issues. The frameworks, which are based on the rough or detailed process approaches, utilize diagrams and verbal descriptions rather than XML in business process issues. If the framework follows the generic process approach, it uses XML to represent a business process in a machine-executable format. The business process faces the same kinds of phases in its automated execution as the business document does in the automated communication.

With regard to messaging issues, the framework may use XML in the headers, which contain data to route the message and to process its attachments and content. This content is always a business document. Other use of XML depends on those standards referred to by the framework.

2.4.3 Comparison

Business document, business process, and messaging issues give a sound basis for comparing different e-business frameworks. The literature provides a number of studies of e-business frameworks (Hasselbring and Weigand 2001, Li 2000, Shim et al. 2000, Zhao and Sandahl 2000). These studies are more descriptive than comparative, and they are already outdated. Novel frameworks have been published, many frameworks have changed, some have ceased to exist, e.g. BizTalk Framework (www.biztalk.org), and some have become inactive, e.g. XML/EDI (www.xmledi-group.org). In addition, there are frameworks that do not use XML, e.g. Open Buying on the Internet (OBI) (www.openbuy.org). Table 1 reviews XML-based e-business frameworks supporting industrial procurement, design, production, or distribution, which have been active after 2001.

Table 1: XML-based e-business frameworks active after 2001

Framework	Site	Initiator	Purpose
AEX (Automating Equipment information eXchange)	www.fiatech.org/projects/idim/aex.htm	National Institute of Standards and Technology	To exchange information for capital facility equipment engineering, procurement, construction, and operations and maintenance work processes
BMEcat	www.bmecat.org	Federal Association of Materials Management, Purchasing and Logistics	To exchange product catalogs electronically
BPEL4WS (Business Process Execution Language for Web Services)	www-106.ibm.com/developerworks/library/ws-bpel	BEA, IBM, Microsoft	To describe business processes and to exchange messages
CIDX (Chemical Industry Data exchange)	www.cidx.org		To exchange business documents in the chemical industry electronically
CITE (Construction Industry Trading Electronically)	www.cite.org.uk	Centre for e-Business in Construction	To exchange business documents in the construction industry electronically
eBIS-XML	www.ebis-xml.net	Business Application Software Developers Association	To exchange orders and invoices between different accounting applications
eBuild-XML	www.basda.org	Business Application Software Developers Association	To exchange orders and invoices between house builders and suppliers
isXML (iron and steel XML)	www.steel.org/xml	American Iron and Steel Institute	To exchange business documents in the steel industry electronically
iXF (interoperable product data eXchange Format)	www.ixfstd.org	SmarTeam	To exchange product data electronically
IXRetail (International XML Retail)	www.nrf-arts.org	Association for Retail Technology Standards of the National Retail Federation	To interface applications within the retailer
MTML (Marine Trading Markup Language)	www.meca.org.uk	Maritime e-Commerce Association	To exchange information in marine trading electronically
OpenTrans	www.opentrans.org	Fraunhofer Institute	To exchange orders and invoices electronically
PDX (Product Definition exchange)	www.pdxstandard.org		To exchange product data between original equipment manufacturers, electronic manufacturing service providers and component suppliers
PIDX (Petroleum Industry Data exchange)	committees.api.org/business/pidx/index.html	American Petroleum Institute	To exchange business documents in the oil and gas industry electronically
PSL (Process Specification Language)	ats.nist.gov/psl/xml/process-descriptions.html	National Institute of Standards and Technology	To exchange process metadata for applications involving discrete processes
STAR (Standards for Technology in Automotive Retail)	www.starstandard.org		To exchange information between automotive dealers and manufacturers electronically
TranXML	www.openapplications.org/downloads/tranxml/tranxml.htm	OAG	To exchange information between shippers and carriers for procurement and delivery of transportation and logistics services
UBL (Universal Business Language)	www.oasis-open.org/committees/ubl	OASIS	A vocabulary based on existing vocabularies

Table 2 shows XML-based e-business frameworks that are compared in more detail in this thesis. cXML, OAGIS, and xCBL are pioneers in XML-based supply chain integration and, therefore, included in the comparison. papiNet and RosettaNet are of interest because Finnish companies are involved in their development. BPML, ebXML, and XPDL, in turn, provide a new insight into the use of XML in supply chain integration.

Table 2: XML-based e-business frameworks compared

Framework	BPML	cXML	ebXML	OAGIS	papiNet	RosettaNet	xCBL	XPDL
Site	www.bpmi.org	www.cxml.org	www.ebxml.org	www.openapplications.org/oagis	www.papinet.org	www.rosettanet.org	www.xcbl.org	www.wfmc.org
Initiated by	Intalio	Ariba	OASIS, UN/CEFACT	OAG	IDEAlliance, AF&PA, CEPI		Commerce One	WfMC
Initiated in	2000	1999	1999	1998 (non-XML 1995)	1999	1998	1997	1998 (non-XML 1993)
Version	1.0	1.2	BPSS 1.0, ebRIM 2.0, ebRS 2.0, ebCPP 2.0, ebMS 2.0	8.0	2.1	About 100 PIPs ready, RNBD 2.1, RNTD 3.0, RNIF 2.0	4.0	1.0
Target	Internal	Cross–industry	Cross–industry	Internal, cross-industry	Paper and forest product	Information and communication technology	Cross–industry	Internal
Based on		ASC X12, EDIFACT	Open-EDI			Open-EDI	ASC X12, EDIFACT	
Documents	No	Yes	Modeling	Yes	Yes	Yes	Yes	No
Number of		34		About 190 as verb-noun-pairs	22	About 120	44	
Validation		DTD		XSDL	XSDL	DTD	XSDL	
Processes	Generic	Rough	Generic	Rough	Detailed	Detailed	Rough	Generic
Process description	XSDL	Request-response and one-way categories	DTD, XSDL	Scenario diagrams	Business scenarios and dialog diagrams	Business scenarios, flow and dialog diagrams	Verbal scenarios	XSDL
Messaging	Web Services		ebMS 2.0	Modified RNIF 2.0	Modified ebMS 2.0	RNIF 2.0		Web Services
Transportation			HTTP/S, SMTP			HTTP/S, SMTP		
Packing			SOAP			MIME/ Multipart-Related		
Signature			XMLDSIG			S/MIME		

The summary of the 18 frameworks in Table 1 gives a rough view. Excluding some exceptions, e.g. CIDX and PIDX, these frameworks fit into the following category:

- *Document-centric frameworks* concentrate on business document issues, ignoring business process and messaging issues. This category is composed of cross-industry and industry-specific vocabularies for business documents.

The comparison of the eight frameworks in Table 2 results in the following categories:

- cXML, OAGIS, and xCBL are *cross-industry frameworks*. They provide cross-industry vocabularies but are limited to the rough process approach and are not greatly concerned with messaging. This category is the oldest one and emphasizes that supply chain integration is mainly about the interoperability of business documents.
- papiNet and RosettaNet are *industry-specific frameworks*. These frameworks give industry-specific vocabularies. However, their main contribution is in business processes because papiNet and RosettaNet provide a comprehensive description of business processes in a particular industry by applying the detailed process approach. In addition, they take into account messaging.
- BPML, ebXML, and XPDL are *process-centric frameworks*: They provide no vocabularies but focus on business processes taking the generic process approach. This category is the newest one, and therefore very fragmented. Only ebXML clearly deals with public business processes between trading partners and puts efforts into messaging and other issues, such as the collaboration infrastructure. In comparison, BMPL and XPDL are newcomers that focus on private business processes within a trading partner. All these frameworks include potential ideas for automated process execution.

2.4.4 xCBL

Common Business Library (xCBL 2003) was started in 1997 by Commerce One, and is a pioneer framework applying XML to e-business. xCBL is a set of XML-based components that allows the creation of XML-based business documents. A vocabulary of the most common cross-industry business documents has been developed after ASC X12 and EDIFACT. Utilizing the EDI semantics, xCBL aims at to speed and facilitate the implementation for existing systems based on EDI. xCBL also strives to preserve and extend the investments made in EDI.

Library

The library of xCBL 4.0 consists of 44 business documents in eight namespaces in addition to the core. These categories are Order Management, Preorder Management, Financial, Material Management, Message Management, Application Integration, Catalog, and Statistics and Forecasting areas. The library comprises Global Elements, which are business documents in a certain namespace. There are also verbal scenarios for exchange of these business documents. In addition, the library includes ComplexTypes, which are components consisting of elements with attributes, and

SimpleTypes, which are enumerations for the elements and attributes. xCBL provides schemas for business documents in the XSDL format.

- Core: This contains components used several times in other namespaces.
- Order management: These documents are used for general order creation and processing. These include any documents exchanged between trading partners for the procurement of goods or services.
- Preorder management: These are used before order creation. These include documents used for confirmation or validation of price and inventory information.
- Financial: These are used for the processing of payment for invoicing the goods or services. This generally includes documents that are exchanged between a trading partner and a financial institution.
- Materials management: These are used for managing inventory. This includes documents associated with the forecasting, shipment, or receipt of goods or services.
- Message management. These are associated with generic xCBL document processing. This includes any documents that are to be used for general acknowledgement, response, and error communication.
- Application integration: These are used to interface with backend ERP systems.
- Catalog: This document is associated with catalog content creation, processing, and inquiries.
- Statistics and forecasting: These documents are used to provide statistical data and forecasting data for products over a specified time period.

Summary

xCBL stems from the eCo framework project (Glushko et al. 1999). xCBL focuses on business document issues although it provides business process information on the exchange of these documents in the form of verbal scenarios. Otherwise, it does not deal with business process and messaging issues. xCBL raises some questions:

- Since xCBL has been modeled after both ASC X12 and EDIFACT, this leads to problems in interoperability due to the differences between ASC X12 and EDIFACT.
- The newest version of xCBL is to some extent incompatible with the older versions.

2.4.5 RosettaNet

RosettaNet (2003) started in 1998 as a consortium of information technology companies for implementing and promoting open e-business standards. It was expanded to include electronic component companies in 1999, and semiconductor manufacturing companies in 2000.

RosettaNet divides the supply chain to eight clusters:

- RosettaNet Support provides administrative functionality.
- Partner Product and Service Review allows information collection, maintenance and distribution for the development of trading-partner profiles and product-information subscriptions.
- Product Information enables distribution and periodic update of product and detailed design information.
- Order Management supports price and delivery quoting, purchase order initiation, status reporting, and management, and order invoicing, payment and discrepancy notification.
- Inventory Management enables collaboration, replenishment, price protection, reporting and allocation of constrained product.
- Marketing Information Management enables communication of campaign plans, lead information and design registration.
- Service and Support provides post-sales technical support, service warranty and asset management capabilities.
- Manufacturing enables the exchange of design, configuration, process, quality and other manufacturing floor information to support virtual manufacturing.

Clusters 1, 2 and 3 extend processes already specified in EDI. Cluster 4 is much like the VICS Collaborative Planning, Forecasting and Replenishment standard (www.cpfr.org). Clusters 5, 6 and 7 provide new kinds of processes. Each cluster is subdivided into two or more segments, each of which comprises several Partner Interface Processes (PIPs). A PIP contains one or more business activities, which in turn specify business actions. In the middle of 2003, there were about 100 PIPs published.

Two steps can present the use of RosettaNet:

1. In the implementation phase a trading partner acquires RosettaNet specifications and downloads the RosettaNet Dictionaries. The trading partners make a trading partner agreement on PIPs, and implement them.
2. In the run time phase a trade partner's RosettaNet-compliant software creates a payload from the system, places this payload into a message, and sends the message as a request to another trading partner. The other trading partner's RosettaNet-compliant software receives this message, opens and validates its payload, wraps the payload in the system, and sends a response according to the choreography of a business process.

Partner Interface Processes

PIP specifications are based on peer-to-peer message exchange between the RosettaNet-compliant software systems. This relies on prior knowledge of the peer identities and their addresses, which should be exchanged by the trading partners in advance. A PIP shows the roles, messages, and their sequence of exchange. It is limited to the public process between the trading partners that trigger the execution or are triggered by the execution of the private processes within the trading partner. The PIP includes the Business Operational View (BOV), Functional Service View (FSV), and

Implementation Framework View (IFV) based on the Open-EDI Reference Model (ISO 1997). The PIP uses Unified Modeling Language (UML) as a notation for flow and dialog diagrams.

The BOV captures the semantics of business data entities and their flow of exchange between roles as they perform business activities. The BOV defines the business process of the PIP, describes its purpose, and specifies its start state and end state conditions. The PIP Business Process Flow Diagram illustrates the business activities and business documents that are exchanged in the PIP. The Partner Role Descriptions describe roles in the PIP. The Business Activity Descriptions describe business activities between these roles. The Business Activity Performance Controls specify the security, audit, and process controls relating to these activities. The PIP Business Data Documents specify business documents that are generated and exchanged by the roles.

The FSV is systematically derived from the BOV. The FSV specifies the network component design and possible network component interactions. A network component design is comprised of agent components and service components that enable the roles to perform business activities. Agents are often implemented as clients and services as servers. A service component can both initiate requests to other services and respond to requests from other services and agents, whereas an agent component can only initiate requests but cannot respond to requests. A browser is a particular type of agent that acts on behalf of a human. The network components collaborate by exchanging business action messages and business signal messages. The Network Component Collaboration specifies the network components and their message exchange. The Network Component Specification specifies the roles in the BOV as network components in the FSV. The Business Action and Business Signal Specification specify business documents in the BOV as business actions in the FSV. The Business Transaction Dialog Specification specifies business activities between the roles in the BOV as network component interactions in the FSV. A network component interaction is a dialog diagram that represents the message exchange when the network components collaborate to execute the PIP. The Message Exchange Controls show the properties for each of the messages exchanged by these interactions.

The IFV specifies the message formats and communication requirements between the network components supported by the RosettaNet Implementation Framework. The Business Message and Communication Specification specify the business messages and their communications requirements by the network component interactions in the FSV.

Each PIP delivers schemas for its business action messages in DTD format. It also delivers the Message Guidelines. The Message Guidelines are structure definitions of the business action messages that outline cardinality of business properties and business data entities, and provide user notes. They provide more complete information than DTDs.

Dictionaries and Codes

The dictionaries ensure the consistency of the information exchanged between the trading partners when executing the PIPs. The RosettaNet Business Dictionary (RNBD) 2.1 defines the business properties, business data entities, and entity instances for the PIPs. The business data entity represents structured business information and is made of other business data entities, whereas the business property is the association between the business data entities. The RosettaNet Technical Dictionary (RNTD) 3.0 provides common properties for defining products for the PIPs. It is a table that organizes product descriptions into reusable atomic properties and relationships and can be used to search electronic catalogs and maintain technical information. However, it is up to the trading partners to develop this content.

RosettaNet uses both the Global Trade Item Number (GTIN), which is a multi-industry standard to uniquely and globally identify goods and services, and the Universal Standard Products and Services Classification (UNSPSC), which allows trading partners worldwide to uniformly classify goods and services. RosettaNet also employs the Data Universal Numbering System (D-U-N-S) that is a worldwide standard for company identification distinguishing unique business locations around the world. These codes are intended to be used with the PIPs.

Implementation Framework

The RosettaNet Implementation Framework (RNIF) 2.0 provides implementation guidelines to create software that executes PIPs. The RNIF specifies how to transport, route, and pack messages between the trading partners. The RNIF supports the use of HTTP/S and Simple Mail Transfer Protocol (SMTP) protocols for the message exchange. A message always contains three headers and the content, which contains a business action message or business signal message. If this content is a business action message, one or more attachments may be included. A MIME/Multipart/Related envelope is used to package the headers and content. S/MIME is used for digital signatures. The RNIF delivers schemas for headers and business signal messages in DTD format. It also delivers MessageGuidlines for these messages.

Summary

RosettaNet covers business document, business process, and messaging issues. RosettaNet contributes a concrete combination of business processes and business documents. It is a good practice that a new PIP has to pass through a secured test or production use for a period to be published. This requires a software implementation. A drawback is that the total number of PIPs is not known for certain. Obviously, the specification of the simplest PIPs is behind but that of the most difficult ones is ahead. In addition, some questions need to be answered:

- How does an individual PIP depend on other PIPs?
- How does RosettaNet enable multi-party B2B interactions?
- How does RosettaNet support transactional management?
- How does RosettaNet ensure that the new versions of PIPs are backward compatible?

2.4.6 ebXML

ebXML (ebXML 2003) started in November 1999 with an 18-month timeframe and the final specifications were finished in May 2001. This framework was officially established by OASIS (Organization for the Advancement of Structured Information Standards) and UN/CEFACT (United Nations Center for Trade Facilitation and

Electronic Business). Its mission was to create a global e-business standard for companies of all sizes.

ebXML is neither industry-specific nor is it merely cross-industry. It aims to be a cross-industry standard that can be developed for use within the industry-specific ones. That is other frameworks, such as RosettaNet, can plug into ebXML. ebXML aims to build on existing standards. It also uses the Open-EDI Reference Model (ISO 1997). The BOV addresses the semantics of the business data in data interchanges and the architecture for business transactions. Its results are Business Process and Information Meta Models for ebXML-compliant software. The FSV addresses the supporting services and focuses on the information technology aspects. The assumption is that software vendors may use the FSV as a reference model to guide them in the software development. Figure 10 gives an overview of ebXML.

Three steps can present the use of ebXML:

1. In the implementation phase, a trading partner acquires ebXML specifications, studies them, and downloads the Core Library and the Business Library. The trading partner either requests the other trading partners' Business Process and Information Metal Model for analysis, or it implements its Business Service by utilizing third party applications. This Business Service should support updates to Core Libraries and Business Libraries. After implementation, this trading partner submits its Collaboration Protocol Profile to Registry Service.
2. In the discovery and retrieval phase, a trading partner requests the Collaboration Protocol Agreement of another trading partner and submits it to this.
3. In the run time, phase messages are being exchanged between these trading partners utilizing the Messaging Service.

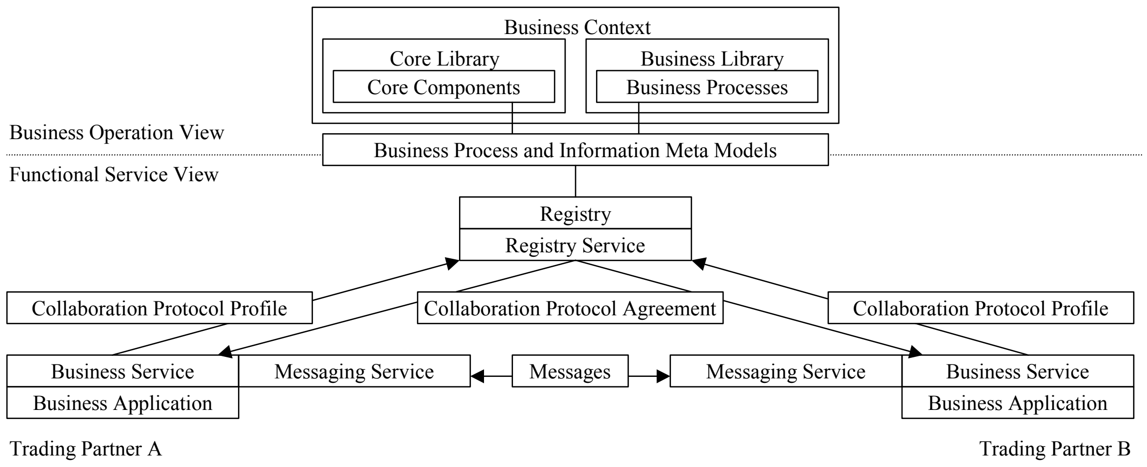

Figure 10: An overview of ebXML

Core Components

Since the exchange of business documents supports the business processes, interoperability requires that the same semantics lead to the same information in different business processes. Therefore, the core components are based on reusable building blocks and the use of context. A component is a building block of business documents that contains information belonging to a single concept, whereas a core component is a generic building block that is context-free. When a core component is used in a business process, it has to be set into a context that is the description of the environment. Each context-specific use of a core component is cataloged under a business information name. A core component is either an atomic block or an aggregate of these blocks semantically completing each other. Atomic core components that naturally fit together can be grouped into aggregate core components. A core component is a common component when it is generic and can be used across several business sectors. Respectively, a domain component is specific to an individual business sector and is only used within that domain. It becomes a common component if it is suitable for use by another domain. ebXML did not deliver a specification for core components. This work is continued by UN/CEFACT and OASIS.

Business processes

Business processes describe how the trading partners take shared roles, relationships, and responsibilities to facilitate interactions with other trading partners. An interaction between the roles follows a choreographed set of business transactions, each of which is expressed as an exchange of electronic business documents. Business process and information modeling is not mandatory but if the users want to model their business processes and information, they should use the UN/CEFACT Modeling Methodology (UMM) that utilizes UML. The UMM metamodel describes the semantics. Since a full UMM model contains more information than is needed to configure ebXML-compliant software, the Business Process Specification Schema (BPSS) 1.0 adopts a subset of UMM for this particular purpose. The BPSS is represented in both UML and XML. It provides information necessary to specify the choreography of interactions between trading partners. In addition, it provides configuration parameters for the trading partner's applications to execute these interactions. The BPSS is intended to be interpretable by ebXML-compliant software. Therefore, the BPSS is available in DTD and XSDL formats

Collaboration Protocol Profiles and Agreements

The ebXML Collaboration Partner Profile (ebCPP) 2.0 provides definitions for a collaboration protocol profile (CPP) that describes the trading partner's business and technical capabilities but excludes the legal terms and conditions. The exchange of information between two trading partners requires each party to know the other party's supported business collaboration and the technical details about how the other party sends and receives messages. A trading partner may have multiple CPPs that describe business collaborations that it supports, e.g. in different regions of the world or different parts of its organization. Respectively, collaboration protocol agreement (CPA) describes the capabilities that two trading partners have agreed to use to perform particular business collaboration. The CPA is an intersection of the trading partners' CPPs that can be processed by these partners' systems to execute the exchange of information. The CPP and CPA are available in DTD and XSDL formats.

Registry

The ebXML Registry Services (ebRS) 2.0 defines the syntax and semantics of the registry services. A registry provides services that enable sharing of information between the trading partners, and maintain the information as objects in a repository.

The registry supports information, such as collaboration protocol profiles, core libraries and business libraries. The object management service provides allows a client to manage the life cycle of objects. The object query management service allows a client to search for a repository entry. The repository entries are stored according to the ebXML Registry Information Model (ebRIM) 2.0. This model does not deal with the content of the repository but represents the metadata of the content. The model can be used to create ebXML-compliant software. Since all objects in the repository have a unique identification, this identification of an object is generated by the repository and may be used by other objects to reference this object.

Message Services

The ebXML Messaging Services (ebMS) 2.0 defines formats and protocols for exchanging messages. In fact, it describes how to exchange XML-based business documents with HTTP/S and SMTP. The specification also covers the definition of a message structure to package payload and the behavior of the message service handler that sends and receives those messages. The message service is defined as a set of extensions to the SOAP and SOAP Messages with Attachments. A message is a MIME/Multipart envelope containing the header container, SOAP message, and zero or more MIME parts of payloads. The header container consists of the SOAP envelope element containing SOAP Header element with ebXML-specific header elements and SOAP Body element with message service handler control data and information related to the payload. XML Digital Signatures (XMLDSIG) is used for digital signatures.

Summary

ebXML deals with business document, business process, messaging, and other issues. However, it approaches messaging issues at a concrete level but the rest of the issues at an abstract level.

The ideas of the BPSS and registry can be regarded as main contributions of ebXML. A drawback is that ebXML specifications are incomplete with respect to essential parts, i.e. core components are missing. Another drawback is that ebXML have been tested only in the proof-of-concept sessions. The software implementations during the specification process would have improved the validity of the ebXML specifications. There are also open questions:

- Does the aggregation of core components make sense if a business document can be a core component consisting of multiple other core components?
- Do multilateral collaborations have to be combined from bilateral collaborations?
- Does transactional management have to be modeled explicitly as part of the choreography of a business process?
- Since the legal aspects are excluded from the CPPs and CPAs, it may be difficult to replace the trading partner agreements.
- It may be easy to store information but difficult to retrieve exactly the information that is needed if the repository offers only a structure of name-value pairs to classify objects.

2.5 Experiences

There seems to be little experiences from XML-based supply chain integration after excluding the software vendors' success stories. The literature provides few scientific papers and public reports. The following experiences are based on ten studies. The first five studies focus on the technical aspects of the prototypes, the sixth study deals with the technical aspects of the operative system, and the last four studies concentrate on the business aspects of the operative systems.

Fürst and Schmidt (2001) present a prototype using XML. This prototype called Data-Extractor was developed for BMW Motors Steyr in Austria to enable that data for controlling the parts delivery is available for all customers, suppliers, and carriers of this factory. The prototype is Java Servlet running on a web server. It handles the requests from the different systems and users working with a browser. The prototype was designed to support warnings of the problems in parts deliveries, reports of the actual status of a parts transport from the carriers, report of the actual parts stock at BMW and the suppliers, report of the incoming of a delivery, and sending of the actual delivery data to BMW. The prototype was tested with simulation data stored in XML on a web server but no connections to the systems at BMW, the carriers and the suppliers were realized.

Sundaram and Shim (2001) present a prototype for RosettaNet. This prototype is a three-tier client-server that allows the customers to order by a browser. The client tier provides a web form using HTML and JavaScript. The business logic tier processes the request from the client tier and sends the response. This tier is implemented by Java Servlet that communicates with the sales and fulfillment application by HTTP. It encapsulates the functionality required to perform the RosettaNet PIPs, and to transform data to XML documents. The business logic tier also uses the database tier for storing and retrieving information. The database contains information obtained from the RNTD. The sales application displays the catalog to browse different products and to query information for these products, and allows to submit a purchase order as an XML document to the fulfillment application. The fulfillment application processes the XML document, sends the order number to the customer, and maintains a detailed profile of each customer and the status of the orders. The administrator of this application can update the order status to either being approved or shipped. Unfortunately, Sundaram and Shim do not report the experiences from the use of their prototype.

Buxmann et al. (2002) present a prototype called SIMPLEX. The prototype uses XML to describe and structure business documents. Based on these documents, it supports the execution of information exchange, the translations between different XML vocabularies, and the integration into in-house systems. The prototype was written in Java using a web server of Apache that was integrated to Tamino XML database. It also uses the XML parser Xerces and the XSLT processor Xalan of Apache. The prototype is capable of handling the following business documents: Invoice, CatalogueQuery, DeliveryConfirm, Delivery Schedule, Forecast, InventoryQuery, MonitoringTransaction, OrderConfirm, Reclamation, SalesOrder, and PurchaseOrder. The supported e-business frameworks are xCBL, OAGIS, and eBIS-XML. Although the prototype was not tested in real cases, Buxmann et al. concluded that the trading partners can use XML as a common language building on it different business vocabularies, and XSLT for translations between these vocabularies.

Lu et al. (2002) present an XML/EDI prototype for Taiwan's flower distribution channels. This prototype server was implemented using Microsoft Internet Information Server, and the XML parser Xerces and the XSLT processor Xalan of Apache. The prototype server using the XML/EDI framework was developed because many farmer associations, retailers, carriers, and wholesalers cannot afford maintaining a server, and there was no EDI standard in Taiwan's agricultural industry. The basic idea was that an outside organization could be in charge of running this server. Farmer associations upload bills of lading to the server daily. This server notifies wholesalers by e-mail, the wholesalers download the bills, process them, and upload the invoices. Then the server notifies the farmer associations by e-mail and these associations download the invoices. Lu et al. do not report experiences from their prototype server in a real case.

Chan et al. (2002) present a prototype for retailing inventory control. This prototype is a two-tier client-server system that was implemented with Java Servlets. In addition, an Inventory Control Markup Language was developed based on XML that is used to construct data models and specify the data exchange format between the client applications and the mediators, and also between the mediators. Common Object Request Broker Architecture (CORBA) was used to implement the communication infrastructure between the mediators. The mediator is based on CORBA objects running on a web server. The prototype provides services for transaction data capture, assortment planning, and automated replenishments. The transaction data capture services are aimed at defining and maintaining basic item information, at recording and issuing purchase orders to the suppliers, at updating information on the received items, at adjusting inventories, and at performing item information lookups, calculating discounts and capturing sales related information. Inventory Control XML defines DTDs for ServiceRequest, ServiceResponse, ResourceAccess, ResourceAccessResult, Buyer, Supplier, Item, and PurchaseOrder documents. Chan et al. provide some test results of the computational performance of their prototype.

StoraEnso North America (papiNet 2002) has implemented the papiNet framework with Time Inc. for PurchaseOrder, Invoice, DeliveryMessage, and OrderConfirmation documents. The system was implemented so that orders received in EDI were sent in parallel via XML. All documents in question were mapped to the order database and an XML document was created when the order became available, paralleling the EDI messages. Time Inc. began by testing purchase orders based on their test scenarios, and StoraEnso North America stored these orders into a database and generated order confirmations. The tests indicate feasibility of the chosen documents.

Avnet (Olson and Williams 2001) has implemented the RosettaNet PIPs 3A4 (Request Purchase Order), 3A6 (Distribute Order Status), and 3B2 (Notify of Advance Shipment) with one supplier of its Computer Marketing Group. The previous implementation was based on EDI. The net present value of cost savings including start-up costs of the PIPs was two million dollars over five years with an interest rate of 11 per cent. These cost savings were realized by eliminating VAN maintenance costs and reducing the order-processing time.

Arrow (RosettaNet 2001) has implemented the RosettaNet PIP 3A4 with UTEC to replace an EDI-based purchase order process. During the first six months of operation, the implementation shows that manual order processing was reduced by 93 to 95 per cent, the inventory turnover rate was doubled, order response time was reduced from 8-

10 hours to 2 hours for exceptions requiring manual intervention and to less than 20 minutes for non-exceptions.

STMicroelectronic (RosettaNet 2002a) has implemented the RosettaNet PIPs 4A4 (Notify of Planning Release Forecast), 4B2 (Notify of Shipment Receipt), and 4C1 (Distribute Inventory Report) to enable collaborative forecasting, inventory allocation and reporting with its customers and key suppliers. This process was manual before the implementation. During four weeks of operation, this implementation resulted in a 50 per cent reduction in contract costs and eliminated 80 per cent of the manual transactions. In addition, a 30 per cent increase in the utilization of capacity was realized.

RosettaNet (2002b) provides a field study covering 12 implementations in the RosettaNet PIPs 3A4, 3A8 (Request Purchase Order Change), 3A9 (Request Purchase Order Cancellation), 3D8 (Distribute Work in Process), 3D9 (Query Work in Process), and 5D1 (Request Ship from Stock and Debit Authorization). In four of the five common tasks, RosettaNet earned greater compatibility than EDI, measured by a five-point scale so that strongly disagree was marked as one and strongly agree as five. RosettaNet also earned more than twice the compatibility rating of semi-automated integration. Cost savings ranged from 16 to 87 per cent. This includes both the implementation and operating costs.

2.6 Critical summary

2.6.1 XML technologies revised

XML is nothing more but a context-free language (Appendix 1). Open source tools, such as Lex and Yacc (Aho et al. 1986), capable of scanning and parsing context-free languages have existed for 20 years. Validation and parsing documents of XML-like languages is possible but requires more programming with these tools than with validating XML parsers. On the other hand, these older tools are capable of processing languages with much richer expressive power than XML. There is a trade-off between the ease of XML and the expressive power of the older tools.

The basic XML technologies enable straightforward wrappers to exchange data as XML documents between the trading partners' databases. However, the use of XML in business documents does not provide much new compared to EDI. XML cannot specify the content of the business document in an unambiguous way. Applying XML to business processes and messaging is something that EDI cannot do. Using XML for describing business processes is an interesting starting point for automated process execution. Unfortunately, there seems to be neither implementations nor open source tools for this purpose. One problem is that DTDs and XSDL schemas are able to validate XML documents against simple rules. There can be situations, in which complex constraints associated with combinations of elements, element content, attributes, and attribute values have to be taken into account. At the moment, the DTDs and XSDL schemas do not support this kind of validation but constraint checks have to be programmed.

2.6.2 Expectations revised

XML has been loaded with expectations. It has clear advantages over HTML and SGML. However, not all the features of XML are comparable with EDI. Some expectations are relevant in web publishing but not in supply chain integration. The following counterarguments point out the fact that one coin has two sides:

- XML is flexible: The ability to define other languages can potentially lead to problems because agreement on a common DTD or schema is not self-evident even in a small user community. Tens of e-business frameworks have been standardized using XML. This indicates that XML can be too flexible for this domain.
- XML is human-readable: If the XML document is indented for full-automated communication, human readability makes no sense. Even in semi-automated communication, it is easy to create quite unreadable XML documents. For example, the element ProNa may mean a product name. What about XML documents created in a different language?
- XML is self-describing: Although DTDs and schemas guarantee a certain amount of validity to XML documents, one may use a DTD, whereas another uses a schema to validate the document. How can it be ensured that the trading partners use the same version of DTDs or schemas?
- XML is structured: There are difficulties to store some characters, e.g. angle brackets, and binary data in XML documents. Since XML is structured text, it may take a lot of memory to store and a lot of time to process this data. The possibility of specifying the contents is not free.
- XML is widespread and inexpensive: Processing data in XML documents does not necessarily stop at validation, parsing, or transformation of the documents but more steps are needed for many applications. For example, storing information in or retrieving it from the relational database is often necessary. The necessary widespread and inexpensive tools for all the steps that process the XML documents are not available.
- XML is platform-neutral and widely supported: Although XML is platform-neutral and widely supported, the applications using XML are not guaranteed to be such. For example, a less-supported application may use a proprietary XML document format.
- XML-based systems have lower costs: Modification of legacy systems is not necessary because middleware can be built to transform data between XML and the native format. However, this does not eliminate the costs but shifts them from the legacy systems to the middleware.
- XML separates processing from content: Although XML separates processing from content, it depends on the developers to ensure that this separation really occurs. For example, if certain elements or attributes require processing that is not supported by the basic XML technologies, these element or attribute names may need to be hard coded into the program.

2.6.3 E-business frameworks revised

XML in itself does not specify what information should be shared, when, and how. E-business frameworks have appeared to standardize business documents, business processes, and messaging between the trading partners. Their aim is to solve the integration problems that XML alone cannot solve in the supply chains. In a sense, the frameworks are a response to the flexibility of XML. Business document and business process issues are the most important ones. For messaging issues, some frameworks rely on a solution of another framework, e.g. ebXML or RosettaNet.

The literature lists tens of e-business frameworks using XML to support industrial procurement, design, production, or distribution. However, only a minority of them have been active in 2002 and 2003. The e-business frameworks are mostly cross-industry and industry-specific vocabularies that include DTDs and schemas for a number of business documents. Unfortunately, these frameworks do not pay much attention to the business processes or messaging. As industry-specific frameworks, RosettaNet and papiNet seem to be the most promising ones. Their conceptualization has been based on a limited set of meanings of terms and models of operation. In addition to the business documents, these frameworks take into account also the business processes and messaging in detail. For RosettaNet, a success factor is that new business documents and business processes have to be implemented and tested before they are approved as a part of the framework. Document-centric frameworks, such as cXML, OAGIS, and xCBL, emphasize cross-industry business documents. They are vocabularies that give simple information about the business processes. Process-centric frameworks, such as BPML, ebXML, and XPDL, are novelties that mainly use XML to represent the business processes. This is something XML can do but EDI cannot. Unfortunately, there seems to exist no open source tools for automated process execution with XML.

Instead of outlining documents from the scratch, the use of existing frameworks reduces the time needed to design and negotiate. However, a large number and new versions of the frameworks cause a problem because transformations between the different frameworks and versions are not free. Complete transformations are even not always possible. The question is which frameworks to support? Divergence has led to an increasing number of the e-business frameworks together with more frequent changes in them. This kind of development jeopardizes adoption of the frameworks. In order to promote XML-based supply chain integration, convergence is necessary.

2.6.4 Experiences revised

The papers dealing with the prototypes are limited to the technical aspects. The prototypes seem to provide the experience that the basic XML technologies for validation, parsing, and transformation are applicable in practice. XML has been used only in business documents. XML in business processes or messaging has not been considered. At least, this was not reported. Many prototypes are intended for semi-automated communication in the point-to-point model. The significance of full-automated communication and the hub-and-spoke model is unclear.

The papiNet case follows much the same line as the prototypes. Respectively, other papers on the operative systems are the RosettaNet cases focusing on the business aspects. The RosettaNet cases represent full-automated communication in the point-to-point model. They provide a number of qualitative results that are difficult to interpret. The quantitative results of the cases shed more light on the business impacts. According to these cases, XML-based supply chain integration seems to result in time and cost savings.

Generally, experience from XML-based supply chain integration is quite limited. The implementations are crucial for evaluating the technical aspects of XML in supply chain integration. However, real cases are also needed to evaluate its business impacts. Considering how much efforts have been devoted to developing XML technologies and XML-based e-business frameworks, it is amazing how few experiences have been publicly reported from implementation in supply chain integration.

3 Case study

This section summarizes a prototype of the XML-based integration system, the CA, implemented and evaluated in an industrial case. First, the section deals with the companies and requirements of supply chain integration in the case. It also presents the structure and configuration of the CA, and the technologies used in the implementation of the CA. Finally, the section presents the objectives and results of the evaluation of the CA in the case.

3.1 Case

3.1.1 Companies

ABB Control has a switchgear factory that is located in Vaasa, Finland. ABB Control is a part of the global ABB group and it produces switchgear assemblies for both internal and external use. The application area of switchgears is power distribution and control. Their purpose is to protect low voltage motors in the process industry. A switchgear order is called a project that consists of functional units. The order may include one switchgear with one functional unit or many switchgears with hundreds of functional units.

ABB Control has an in-house information system called COPIS (Customer Oriented Process Information System) that manages many aspects of the order-delivery process related to a switchgear order. The functionality of COPIS covers engineering design, product configuration, material requirements planning (MRP), production planning, quotations, and order tracking. It makes quotes and production resource predictions based on labor requirements, however, activity-based costing is being implemented.

Many of ABB Control's customers require switchgear orders at short notice. The company policy is to deliver special customer tailored solutions by the requested day. ABB's switchgear factory in Vaasa often works very close to or over full capacity and over the last few years the final delivery dates have not been met in some orders. Production planning with a short delivery time is difficult.

The order-delivery process related to a switchgear order is quite traditional in ABB Control. The customer sends a tender request to sales persons or the sales department. The planner feeds the technical data of the tender request into COPIS and configures the switchgear unit. After that, COPIS develops drawings and the tender is sent to the customer. If the customer accepts the offer, all the data and information are available from COPIS.

Using COPIS, ABB Control has visibility of the present factory loading and anticipated loads. When the loads threaten to exceed the factory capacity, the management has looked for alternative production resources in order to clear the overload. ABB Control has several methods for outsourcing the production. These are local ABB outsourcing, i.e. ABB companies in Finland, global ABB outsourcing, i.e. ABB companies world wide, and subcontracting.

InCap Electronics in Vaasa is one of ABB Control's subcontractors providing production resources for switchgear assembly. It is also a subcontractor of, e.g. ABB

Drive and ABB Industry. Although the InCap Electronics' factory is located inside the ABB Control's factory in Vaasa, InCap Electronics and ABB Control use EDI to exchange some business documents.

3.1.2 Requirements

The basic requirement was to continue using COPIS but expand its functionality relative to production management and purchasing with respect of the internal and external production resources. Internal production resources mean ABB Control's own capacity, whereas external production resources mean capacity of other ABB companies or subcontractors. The combination of internal and external production resources forms a supply chain. The prototype integration system extends the functionality of COPIS to support supply chain integration.

The idea behind the integration is that the flexible transparency of the internal and external production resources. With this transparency view to the production resources, the production management has possibilities to plan external resources as internal ones. For ABB Control, the expected business benefits through the supply chain integration are related to the growing production flexibility and more profitable data exchange. Figure 11 illustrates a cooperation model in which a subcontractor produces ABB-specific products, whereas a supplier does not.

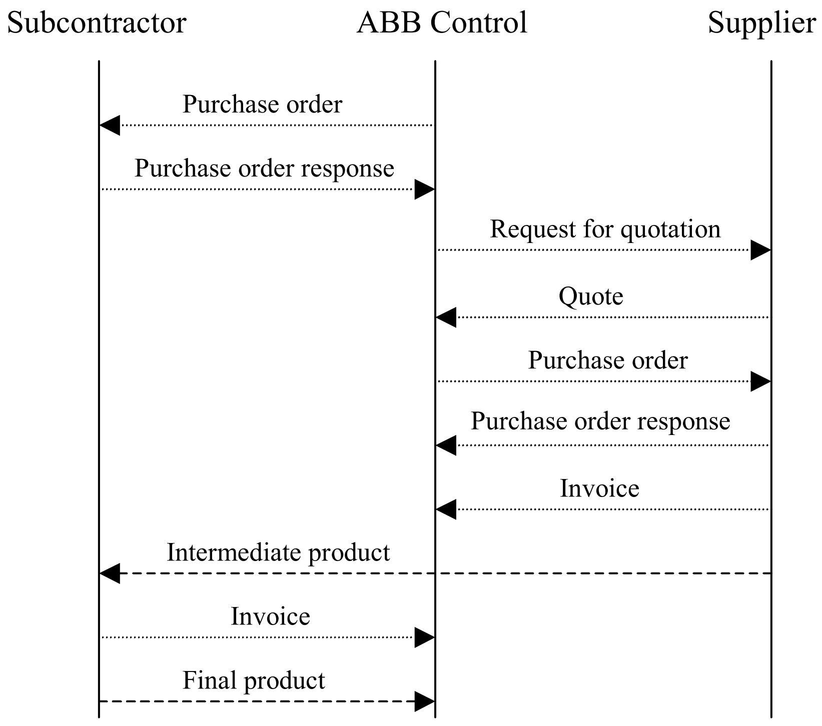

Figure 11: A cooperation model

This cooperation model of supply chain integration could be implemented with the current software technologies. A number of commercial integration systems (e.g. Microsoft 2003) were available. However, taking into account the unstable status of the new technologies, the case gave an opportunity to design and implement a prototype integration system. The objective was to achieve a prototype that can be easily maintained and used, and to utilize XML-based technologies. The purpose was for

XML to be used to extend supply chain integration to those subcontractors who had not accepted EDI. Perhaps later XML will be used to replace EDI.

The architecture of the prototype integration system was based on the model, in which a server supports EDI, XML or both. The clients of this server can be browsers, other integration systems, or applications. This was not the only possible model but many commercial integration systems at least partially conform to this architecture. The implementation of supply chain integration required the design of the required messages and their interaction-handling logic.

A large number of potentially useful interactions were identified in the context of supply chain integration (Kalliokoski et al. 2000). Some of these interactions were so primitive that they are not business transactions. In the case, the objective was that the prototype integration system would be capable of performing the following interactions:

- ABB sends a specific purchase order to a specific subcontractor. Query the database for the EDI segments of a purchase order, translate the purchase order from the EDI segments to the xCBL format, send this purchase order in the xCBL format as an e-mail or a message to the subcontractor, and a return success/failure status.
- A subcontractor queries for one of its purchase orders by a browser. Query the database for the EDI segments of a purchase order, translate the purchase order from the EDI segments to the xCBL format, and return the purchase order.
- A subcontractor queries for a list of all its purchase orders by a browser. Query the database for the EDI segments of purchase orders, translate the purchase orders from the EDI segments to the XML format, and return a list.
- A subcontractor updates a purchase order response in the xCBL format into a database of ABB by a browser. Translate the purchase order response in the xCBL format to SQL INSERT statements for the EDI segments, update the database with the EDI segments, and return a success/failure status.
- A subcontractor updates an invoice in the xCBL format into a database of ABB by a browser. Translate the invoice in the xCBL format to SQL INSERT statements for the EDI segments, update the database with the EDI segments, and return success/failure status.
- A subcontractor queries for a demand forecast of all its possibly forthcoming purchase orders by a browser. Query for possibly forthcoming orders, translate the possibly forthcoming purchase orders from the EDI segments to the XML format, and return a demand forecast.

All these interactions without the last one were used as a starting point in the design and implementation of the prototype. ABB Control suggested them. The last interaction was identified after the implementation and suggested by InCap Electronics. It gave an opportunity to evaluate the maintainability of the prototype.

The content of the purchase order, purchase order response and invoice documents was based on one XML-based e-business framework (ebXML 2003, RosettaNet 2003, xCBL 2003). xCBL 2.0 was utilized in the case because it provided the corresponding XML-based business documents and was interoperable with EDI. XSLT was needed to make the necessary transformations between XML and EDI formats.

3.2 Implementation

3.2.1 Architecture

The prototype conforms to a layered software architecture that could be described as engine-processor architecture. The motivation for this type of architecture is maintainability. The CA is integrated to other systems using standard techniques, such as HTTP, Open Database Connectivity (ODBC), and SMTP. Figure 12 illustrates the engine-processor architecture of the CA.

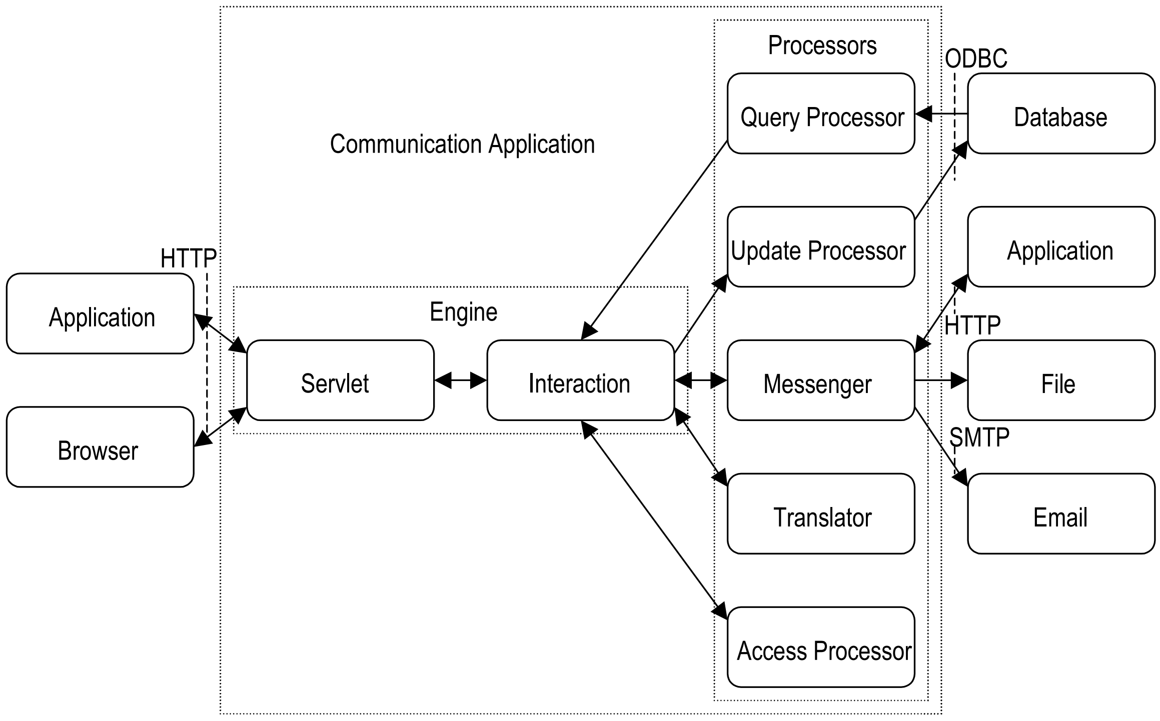

Figure 12: Architecture of the CA

An engine that processes the interaction requests and executes them according to the configured interaction definitions forms the top layer of the architecture. These definitions describe the interaction-handling logic in terms of parameters and operations. The bottom layer of the architecture contains a set of processors that are able to perform the operations. The CA can load these processors and their configuration data "on demand" from the local file system or the Internet. This makes it possible to maintain the functionality of the prototype without code changes to the engine. In addition to portability, Java supports the previous kind of extendability. The system configuration data has to reside on the same host as the system itself whereas all other configuration data can be retrieved from the Internet.

3.2.2 Engine

The engine of the CA processes the interaction requests according to their configuration data. The processing logic is as follows:

- A *Servlet* processes the interaction requests from the clients. The requests can be in either HTML or XML format. According to this request, the Servlet passes the call to an *Interaction*, which is associated with the configured interaction definitions.

- The Interaction executes the requested interactions with the given parameter values and the configured interaction definition as inputs. It interprets the XML-based interaction configuration language and calls the processors according to the configured definitions. When a processor has executed an operation successfully, it saves the output and the Interaction object passes it to another processor if necessary. When all operations of the interaction are executed, the Interaction returns the result.
- The Servlet sends the result to the client as an interaction response. The response can be presented in different formats, e.g. plain text, XML or HTML.

The engine also keeps a log on the executed interactions. For each interaction request, it records information about when the request was executed and from which address it was made. In addition, the engine records the parameter values and result of the interaction. If the interaction fails, an error message is recorded.

3.2.3 Processors

In the case study, the CA has five processors. Since the CA is designed to be expandable, it is capable of loading and calling an object as a processor if the object implements a generic processor interface. The roles of the processors are presented in the following:

- The *Query Processor* retrieves data from a relational database via ODBC. Its inputs are values associated with the SQL SELECT statement to be executed. The output is a result of this statement translated into XML format.
- The *Update Processor* manipulates the data content of a relational database via ODBC. The input consists of SQL UPDATE, INSERT, or DELETE statements in XML format. The processor has no output.
- The *Messenger* saves a document or sends a request to a specified target. Its inputs are either a document or request and a target name. In the case of a request, the Messenger is left waiting for a response, which is its output. The request can be an interaction request and the target may be another integration system. In this way, it is possible to delegate the processing of interactions between integration systems in the different sites.
- The *Translator* is an essential processor from the point of view of supply chain integration. It transforms input from one XML format into another. The output of the processor is a result of the XSLT translation specified as a part of the configuration data.
- The *Access Processor* checks the password given by the user. Its inputs are a user name and password. If the password is invalid, the processor aborts the execution. Otherwise, the output of the processor is the identification associated with the user that can be used by other processors.

The generic processor interface (Appendix 2) is based on methods for initializing a processor with configuration data, setting its input values, executing the processing, and getting its output value. If an object implements the interface, it is utilized as a processor. This is important because the engine is able to interact with the processor by only these methods.

3.2.4 Configuration

A large part of the actual functionality of the CA is defined with XML-based configuration languages instead of lower level programming languages, such as Java. This is motivated by easier maintainability. However, the element or attribute names in DTDs used to validate configuration data cannot be changed because DTDs do not support some checks, e.g. if a particular element with a particular attribute value has occurred already. Therefore, some changes can be done in run-time by modifying DTDs but a lot of information related to validation of the configuration data has to be hard coded in compile-time. Although XML-based configuration languages do not remove the problems of maintenance, they facilitate it being without graphical user interfaces that are often expensive to develop.

The configuration data is divided into three levels. At the *system level*, the configuration data (Appendix 3) specifies which interactions are defined for the system. At the *interaction level*, the configuration data (Appendix 4) defines which parameters and operations are required for execution of an interaction. At the *operation level*, the configuration language is specific to the processor that executes the operation (Appendix 5, 6, 7, 8).

Interaction definitions form the core of the configuration of the integration system. A balance between flexibility and simplicity has been aimed for by defining the interactions as operation block models. The interaction has parameters the values of which are given by the sender of the request. These parameters are inputs for operations that are sequentially executed in the order of appearance. Each operation has a link to the processor that executes the operation with the given configuration and inputs. Some operations return output that works as input for the following operations or as a result of the interaction. If an exception is detected during execution of the interaction, it will be reported to a receiver and the execution will be aborted.

The following example shows the configuration of the CA at the system and interaction level. In this example, a user from ABB Control wants to get information on a purchase order. Figure 13 shows an interaction request (Appendix 9) sent by an application at ABB. This interaction request could also be sent through a homepage (Appendix 10) by a browser.

```
<interaction-request operation="query" format="xcbl"
  document="order" presentation="xml">
  <parameter name="user">
    abb
  </parameter>
  <parameter name="password">
    control
  </parameter>
  <parameter name="order-id">
    123
  </parameter>
</interaction-request>
```

Figure 13: An interaction request in XML

Figure 14 shows the configuration data at the system level. An interaction is identified by an operation-content-document-triple that refers to the interaction definition. The example demonstrates a query-xcbl-order-interaction whose definition is located in file://query-xcbl-order.xml.

```
<configuration dtd="file:request.dtd" html="file:xml-to-html.xsl">
  <interaction operation="send" content="xcbl" document="order">
    file:send-xcbl-order.xml
  </interaction>
  <interaction operation="query" content="xcbl" document="order">
    file:query-xcbl-order.xml
  </interaction>
  <interaction operation="receive" content="xcbl" document="invoice">
    file:receive-xcbl-invoice.xml
  </interaction>
</configuration>
```

Figure 14: An engine configuration

Figure 15 illustrates an interaction definition that has both parameters as input and a result as an output. In the example, parameter values user, password, and order-id are given by an interaction request. The first operation, access, has configuration data that is located in file://query-xcbl-order-access.xml. The program code of this operation is located in file://AccessProcessor.class. The access operation checks the username and password. If they are valid, the execution proceeds to the second operation, query. This operation retrieves the EDI segments for a purchase order with a value order-id from a relational database and represents the result as a result set in the XML format. The third operation, translation, transforms the output of the query operation into the xCBL format. Finally, the interaction returns the output of the translation operation that is transmitted to the user as the interaction response.

```
<interaction-definition>
  <parameter name="user" type="String"/>
  <parameter name="password" type="String"/>
  <parameter name="order-id" type="String"/>
  <operation name="access"
    processor="file:AccessProcessor.class"
    configuration="file:query-xcbl-order-access.xml">
    <input name="user"/>
    <input name="password"/>
  </operation>
  <operation name="query"
    processor="file:QueryProcessor.class"
    configuration="file:query-xcbl-order-database.xml">
    <input name="order-id"/>
  </operation>
  <operation name="translation"
    processor="file:Translator.class"
    configuration="file:query-xcbl-order-translation.xsl">
    <input name="query"/>
  </operation>
  <result name="translation"/>
</interaction-definition>
```

Figure 15: An interaction definition (query-xcbl-order.xml)

At the operation level, the configuration data can be presented in many different ways. Although a processor may have very sophisticated needs considering configuration, it is recommended that the configuration data is represented in XML.

From the viewpoint of maintainability, the hierarchical structure of the configuration data has an advantage by providing independence between the different levels. For example, if the passwords of the users are changed, there is no need to make changes at the system or interaction level. On the other hand, if the access check is removed, the configuration data remains the same at the system level.

3.2.5 Technical details

The implementation of the CA is based on the following open source tools:

- Java 2 platform, standard edition, version 1.3.0 (Sun 2000a)
- Jakarta-Tomcat servlet container, version 3.1 (Apache 2000)
- JAXP (Java API for XML Processing) parser, version 1.0.1 (Sun 2000b)
- Saxon XSLT processor, version 5.4.1 (Key 2000)

The engine of the CA is Java Servlet that is running on Jakarta-Tomcat, a servlet container. A servlet container is not a web server but a runtime shell that manages and invokes servlets on behalf of the users. The engine is implemented by using Java Servlet API. Since the engine loads and calls the processors of the CA dynamically, the communication between the engine and processors is based on Java Reflection API. Where XML is said to provide data portability, Java provides code portability. Moreover, Java Servlet API and Java Reflection API enable accessibility and extendability of the CA in the Internet.

The Query Processor and Update Processor use JDBC (Java Database Connectivity) API to communicate with a relational database via ODBC. The engine, Access Processor, and Messenger use JAXP to validate and parse XML documents according to the SAX approach. In comparison, the Query Processor and Update Processor use a self-made parser to process configuration data. The engine and Translator use Saxon to perform XSLT translations.

3.3 Evaluation

3.3.1 Objectives

The main objective of the evaluation was to find out how and to what extent the CA could support supply chain integration between companies. An objective of the study was to assess the viability of the chosen approach by identifying its potential benefits and expected implementation and operating costs in a supply chain context.

The evaluation plan was originally drafted on the basis of a set of assumptions about the main benefits of XML-based integration systems, such as the CA. The main hypothesis was that they could provide a good basis for supply chain integration between companies. In particular, it was assumed that they would be more flexible to maintain, and less expensive to implement and operate than most currently used B2B systems, such as EDI.

On the basis of these assumptions, it was considered that SMEs especially would be interested in the integration system, provided that its benefits can be identified and illustrated in a proper way. In consequence, it was also considered that even larger companies might regard it as a feasible extension to their existing communication solutions when collaborating with SMEs. An objective of the evaluation process was to assess the adequacy of these assumptions.

The main challenges of the evaluation were related to the identification of those factors that might add either to the cost or to the benefit side of the calculation. Finally, six groups of factors were identified as important contributors to the viability of the integration system concept by evaluators after discussions with developers and representatives of the case companies. The following key areas were given special emphasis in the evaluation:

- *Functionality and technical feasibility*: This area of the evaluation aimed at assessing the applicability of the chosen mechanism to mapping structured business documents of diverse syntax to each other. In addition, attention was paid to the functioning of the implemented interactions as well as to the overall technical feasibility of the prototype.
- *Scope of interactions*: This area was more theoretical in nature and related to the expressive power of the chosen architecture and its configuration mechanism. The objective was to determine the extent to which they could possibly cover ABB’s communication needs in subcontracting and related collaborating.
- *Impact on business*: The objective was to identify the potential business benefits as well as the expected disadvantages of operating a CA type of integration system at ABB. Areas, such as flexibility, reliability, efficiency, speed, and

costs, i.e. the key attributes of any process or practice, were addressed during the evaluation.

- *Implementation demands on the organization*: The objective was to identify the challenges and needs for change that the implementation of a CA type of integration system might cause in ABB's processes, practices and organization.
- *Configuration and maintainability*: The objective was to determine the basic requirements for the definition and modification of interactions in terms of the necessary knowledge and work inputs. Another related objective was the specification of a feasible model and arrangements for the initial configuration process and the related systems support.
- *Use of EDI*: The objective was to examine the utilization of EDI at ABB, to identify its main benefits and support for purchasing practices, and to construct an extensive picture of all EDI-related implementation and operating costs. The purpose of analyzing the use of EDI was to establish a kind of reference point for the CA.

3.3.2 Results

Functionality and business benefits

The implementation process of the CA appeared to be flexible due to the fact that interaction definitions are stored in text files that can be created and maintained by means of ordinary text editors. Functional tests of the implemented interactions were completed successfully, including those that were run over the Internet. However, with regard to purchase order, purchase order response, and invoice the content checks revealed some disparities between the corresponding business documents in xCBL and EDI formats. The identified deviations mostly related to differences between the two formats, which actually made a perfect match impossible. Compared to EDIFACT, xCBL 2.0 did not make the difference between the net and gross price in the purchase orders and invoices. xCBL 2.0 also required information about the document language. This indicates the disparity of xCBL 2.0 and EDIFACT. Such problems can usually be resolved because data structures can be flexibly added to or removed from XML-based business documents.

The chosen engine-processor architecture together with its XML-based configuration mechanism proved to function well and to form a good basis for further development of XML-based integration systems. For example, all interactions that were identified as potentially useful by ABB were either implemented or could have been implemented in the CA. Some of those interaction types were not supported by EDIFACT, and only a few of them were actually used in Finland. Therefore, the test case would suggest the feasibility of the chosen architecture and configuration mechanism as an implementation approach to the integration system.

The test case suggests that the business benefits of integration systems in general are highly dependent on the scope of their implementation and on case-specific needs. In the case of ABB, they mostly related to production planning and supervision, and functioning of the customer interface. In consequence, ABB was especially interested in the possibility of exchanging subcontractor capacity and information related to the order-processing status. The assessed benefits related to enhanced information management, more efficient use of available production resources, and consequently, shorter lead times. InCap emphasized the importance of demand forecasts as a medium in its production planning practice. With regard to purchase orders, purchase order responses and invoices, the expected benefits were similar to those of EDI: higher efficiency of administrative routines.

Demands and costs of implementation and operation

When assessing the potential starting-points for implementing the CA, two different scenarios were identified: establishing a new electronic connection between ABB and one of its subcontractors, or replacing an existing EDI connection. The general conclusion was that in both cases the implementation demands on an organization would be very similar to those of any other information system, i.e. involving extensive negotiations and cooperation with the two partners as well as a great deal of specification and testing. However, some special characteristics were identified.

The first relates to the utilization of the Internet instead of tailor-made connections. As a result, there would be no mediator between the two organizations, which in turn would probably result in significant savings in the communication charges. On the other hand, in such a situation it is up to the end-user organizations to establish proper arrangements for managing the connection, e.g. ensuring data security and eliminating duplicate messages. Usually the EDI operator provides this service. It is not clear what the most feasible future solution to this particular challenge would be and how much it might cost.

Another special characteristic has to do with the case when the subcontractor decides to operate the system through a browser instead of an integration system of its own. In such a case, the subcontractor's implementation process would be relatively straightforward. From the main contractor's point of view, this scenario involves several uncertainties, however. For example, the number of erroneous messages is likely to increase in case of a growing number of manual working stages involving direct database operations. Such errors are likely to end up in operative information systems, thus causing excess administrative work. There is no obvious solution to this challenge although proper training is certainly a key issue here. However, it can be reckoned that the main contractor would have to invest more time and resources in supervising the use of the system.

Configuration of the CA requires a lot of background information. Getting that information requires work and takes time. In the test case, the total amount of time needed for defining one interaction was reported to range from a couple of hours up to a couple of weeks. The large variation mostly relates to the complexity of XSLT translations determined by the characteristics of the data structures to be mapped to each other. A rough estimate was that about 80 per cent of the XSLT code is easy to write, whereas the remaining 20 per cent could require considerably more effort at least in the learning phase. The translation is often considerably alleviated by splitting it into a set of consecutive translations. It was estimated that on average the interaction definition takes around one week of work. In some cases, only about five per cent of the time was actually spent on writing the interaction definitions. These are based on the developers' estimate provided that the work is done by an external expert with a good knowledge of XML, XSLT, and xCBL, with a profound understanding of the integration system, and with no previous knowledge of the company information system to which the integration system is to be connected. These conditions applied to the evaluators and developers of the CA. In addition, the data contents have to be specified in advance, as they were in the test case apart from one interaction. If the configuration process involves business planning, e.g. negotiations between several partners on the content of the messages, much more time will be needed.

In the course of time, things get easier due to the learning effect. For example, modifications to the existing interactions were effected swiftly. The existing interaction definitions also formed a good basis for the definition of new ones of a similar kind.

Comparing the costs of EDI and CA

The implementation costs of EDI were assessed to be clearly higher than the costs of the CA. With regard to the amount of the necessary work alone, the introduction of a new message type that corresponds to the definition of a new interaction in the CA, was found to require a new EDI module and about 200 hours of related specification and testing work. Opening a new connection, i.e. taking a particular message type in use between two partners was found to require a couple of days of testing. Provided that the testing times are about the same with regard to EDI and CA it can be assumed that establishing an EDI connection for a new message type is at least three to four times more expensive than implementing a new interaction in the CA. This figure was based on ABB system specialists' experience on EDI development.

Finnish software firms have shown interest in further development of the CA. They have not been able to give any estimate of the order of the future market price (license fees) for a CA type of commercial integration system. When the charges for use are taken into account, the XML-based integration system seems certainly less expensive because it is based on the use of the Internet. However, it is hard to say whether the application service provider concept will also make progress in this field. Should this happen, could the charges for communicating through an XML-based integration system be commensurate with those of EDI, especially if the related services are provided by EDI operators? This remains to be seen.

4 Discussion

This section compares this research with previous research, discusses its results, presents opinions about XML-based supply chain integration, and presents questions for further research.

4.1 Review discussed

Studies of e-business frameworks (e.g. Hasselbring and Weigand 2001, Shim et al. 2000) are to a large extent outdated, which reflects the rapid development of the frameworks. These studies are simply descriptive and they do not analyze systematically the frameworks to identify their features or to classify them.

A review presents three models for supply chain integration. The point-to-point model reflects EDI thinking. It is still feasible. The hub-and-spoke model is very much a result of the Internet. It often takes the form of an electronic marketplace. Finally, the service-oriented model represents the XML era but its real implementations are missing.

The basic XML technologies enable straightforward exchange of data as XML documents between the trading partners' systems. XML has clear advantages over HTML and SGML. Expectations on XML suggest that XML is flexible, human-readable, self-describing, structured, widespread and inexpensive, platform-neutral and widely supported, and it has low costs, and separates processing from content. Not all the features of XML that are important in web publishing are relevant in supply chain integration. The first impression is that XML is much ado about nothing. The context-free languages can be scanned and parsed using well-known open source tools. However, transformation and simple validation are easier with XML because less programming is required. Unfortunately, DTD and XSDL are sometimes insufficient. Then, the validation may require parsing with a lot of programming. There seems to be many XML technologies under development, e.g. Web Services and Semantic Web. Is it reasonable to develop new XML technologies despite the fact that existing XML technologies suffer from deficiencies? The new versions and XML technologies can even complicate the selection of technologies used in supply chain integration and, therefore, slow down the adoption process.

XML-based e-business frameworks solve integration problems but their large number and frequent versioning also cause problems. Although e-business frameworks reduce the costs needed to agree on interoperability issues, interoperability between different frameworks or versions can be incomplete and the adoption of the framework or its version is not free. Messaging issues are in a mature state but there is much to do with business document issues and business process issues. The basic problem is to jump from revolution to evolution. This requires convergence of the frameworks instead of their divergence, and a reduced number of them. At the moment, industry-specific frameworks seem the most credible because they have a clear focus and industrial usage. Especially, RosettaNet has required in its standardization that novelties have to lead to implementations. This may encourage also other industries to adopt RosettaNet.

This review succeeded to identify the basic features and categories of XML-based e-business frameworks. It also provided a critical summary of the basic XML technologies, expectations and experiences associated with XML-based supply chain integration.

4.2 Case study discussed

There is not much experience from XML-based supply chain integration. Considering how much efforts have been devoted to developing XML technologies and e-business frameworks, it is amazing how few experiences have been properly reported from the prototypes and operative systems. In addition, none of the reported studies (e.g. RosettaNet 2002b, Sundaram and Shim 2001) cover both the technical and business aspects of supply chain integration.

The case study focuses on a prototype of the XML-based integration system that was implemented using Java language and open source tools. The configuration of the prototype that is divided into three layers can be done solely with XML and its functionality that is based on processors can be updated dynamically by an engine. XML is not used only for business documents but also for processes. Instead of business processes, "technical" processes are represented by XML. These processes define the operations and their inputs, outputs, and configuration data in interactions, when the engine executes the processors as operations. In addition to the engine-processor architecture and XML-based configuration mechanism, XSLT for transforming XML documents and xCBL as an e-business framework were a matter of interest.

Instead of using a commercial integration system, design and implementation of the prototype shed more light on the capabilities and limitations of XML and the basic XML technologies although a number of important security and reliability features were not taken into account. Since purchase orders, purchase order responses, and invoices are the most common EDIFACT message types in Finland, the case study covered them. The xCBL 2.0 framework was chosen. It is a pioneer of XML-based e-business frameworks that is based on the EDIFACT standard. However, the case study revealed disparities between the corresponding business documents in xCBL and EDIFACT. For example, the xCBL documents did not make the difference between net and gross prices. These problems can be easily resolved by adding new structures to these business documents. Unfortunately, this causes new problems because the extended business documents are no longer compatible with the framework. In all, incomplete frameworks do not guarantee interoperability.

This case study was successful because it took a stand to both the technical and business aspects of XML-based supply chain integration. Although the result of the case study cannot be generalized as well as the results of the survey, it gives a deep understanding of XML-based supply chain integration.

4.3 Further research

Comparison between XML and EDI is problematic. Traditional EDI-based integration has been implemented with an older programming language, e.g. Cobol, over a tailor-made connection, e.g. VAN, whereas XML-based integration often employs Java over the Internet. If the effects of the programming language and connection are eliminated, what advantages and disadvantages do XML-based e-business frameworks have over EDI standards in supply chain integration? This is the first question for further research.

There are hopes that Web Services and Semantic Web standards will facilitate supply chain integration (Ding et al. 2002, Staab et al. 2003). Therefore, it would be important to analyze the effects of SOAP, UDDI, WSDL, RDF, RDFS, and OWL on supply chain integration. How they do support supply chain integration? This is the second question for further research.

5 Conclusions

Supply chain integration means information sharing between trading partners. These partners may be different companies or units within a company. If the trading partners use different kinds of documents, processes, or systems, it is difficult for them to communicate efficiently. This costs both money and time.

The trading partners must have a shared understanding of their ways of doing business before their systems are able to communicate. The trading partners have to know what information should be share, when, and how. XML is a format for representing documents and process but it does not define any particular document or processes. Therefore, e-business frameworks are necessary for information sharing. The thesis identifies that the frameworks cover the business and technical aspects of business documents, business processes, and messaging. In addition, the framework may take the rough, detailed, or generic approach to the business processes. XML has an important role in the business documents and messaging but only the generic approach applies XML to the business processes.

Analyzing the XML-based e-business framework, it is possible to identify four categories. Most frameworks are document-centric concentrating on cross-industry or industry-specific business documents. Cross-industry frameworks provide business documents and rough business processes across industry, whereas industry-specific frameworks describe business documents and detailed business processes in a particular industry. Finally, process-centric frameworks focus on generic business processes.

This thesis presents a prototype of the XML-based integration system called the CA that was implemented using Java and tested at ABB Control and InCap Electronics. It was used in an industrial case for studying the properties of the XML-based integration systems from both the development and usage perspectives. The development perspective focused on experiments with some implementation approaches to the prototype. Respectively, the usage perspective concentrated on evaluation of the potential benefits and costs of the prototype in an industrial case. Since ABB Control and InCap Electronics used EDIFACT in supply chain integration, it was possible to compare XML-based integration to EDI-based integration.

The prototype exchanged purchase orders, purchase order responses, and invoices based on the xCBL framework, as well as demand forecasts over the Internet successfully, whereas EDIFACT did not support the exchange of demand forecasts. Since XML enables customized business documents, which can be validated by DTDs, parsed by SAX, and transformed by XSLT, and the integration systems are based on the Internet instead of VAN, the prototype was more flexible to implement and operate than EDI. In addition, the engine-processor architecture together with its XML-based configuration facilitates the maintenance of the prototype. Therefore, the prototype provides a sound basis for XML-based supply chain integration.

At this point, only preliminary results concerning the industrial use of the prototype exist. The results show that within the test case the prototype can fulfill the functional requirements of supply chain integration. The business benefits of the prototype are highly case-specific but its use could provide significant cost savings in comparison to EDI. However, it is not completely clear to what extent these savings are attributable to the properties of the prototype or to other contributing factors, e.g. how EDI operators do price their services. An EDI connection for a new message type was three to four times more expensive to implement than a new message type with the prototype. Therefore, the implementation costs seems to be lower for XML than for EDI. On the other hand, a subcontractor needs no integration system of its own but it can use the main contractor's XML-based integration system by a browser. For SMEs requiring less frequent data exchange, this is an important factor. When charges for VAN are compared to those for the Internet, the operating costs seems to be lower for XML than for EDI. This is not necessarily a permanent situation because more intense competition between EDI operators may bring down charges for VAN.

There are more XML-based e-business frameworks than EDI standards. On the other hand, EDI cannot represent business processes, which XML can do. This indicates that XML is more flexible than EDI. The same coin has another side. Which framework to support, if any? Adoption of a large number of frameworks is more expensive than adoption of a small number. Choosing a framework that matches with the trading partners tends to be more difficult from a large number of frameworks than from a small number. Experiences also indicate that XML-based integration is less expensive than EDI-based integration. These indications are tentative due to the lack of accumulated experiences.

In the short run, XML and EDI are more likely complements than substitutes. Large enterprises will not abandon EDI in many industries until the uncertainty about XML is dispelled. However, XML provides large enterprises a way to extend supply chain integration with SMEs, which have no legacy systems and are thus unwilling to invest in integration systems. The idea is that a SME has an e-mail address and access to the Internet, and a large enterprise has an integration system to be used over the Internet by a browser.

In the long run, XML-based supply chain integration can be a significant alternative to EDI although XML does not remove all the integration problems. Shared understanding of business documents and business processes is still necessary in the future. XML does not guarantee but it can promote achieving a shared understanding that is needed in supply chain integration.

References

Aho, A., Sethi, R., Ullman, J., 1986, *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, Reading, MA.

Anandarajan, A, Wen, H., 1999, "Evaluation of information technology investment", *Management Decision*, Vol. 37, No. 4, pp. 329-337.

ASC, 2003, *ASC X12*, http://www.x12.org

Apache, 2000, *Jakarta-Tomcat*, http://jakarta.apache.org/tomcat

BEA, 2003, *WebLogic Server*, http://www.bea.com

BizTalk, 2002, *BizTalk Framework*, http://www.biztalk.org

BPMI, 2003, *Business Process Modeling Language*, http://www.bpmi.org

Buxmann, P., Díaz, L., Wünstner, E., 2002, "XML-based supply chain management: As SIMPLEX as it is". In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, Vol. 7. HICSS 02, January 7-10, Big Island, HI. Pp. 2141-2150.

Chan, S., Dillon, T., Siu, A., 2002, "Applying a mediator architecture employing XML to retailing inventory control", *Journal of Systems and Software*, Vol. 60, No. 3, pp. 239-248.

Copeland, K., Hwang, C., 1997, "Electronic data interchange: Concepts and effects". *The 7th Annual Conference of the Internet Society*. INET 97, June 24-27, Kuala Lumpur, Malaysia.

cXML, 2003, *Commerce Extensible Markup Language*, http://www.cxml.org

Cummins, F., 2002, *Enterprise Integration: An Architecture for Enterprise Application and Systems Integration*. Wiley, New York, NY.

Ding, Y., Fensel, D., Klein, M., Omelayenko, B., 2002, "The semantic web: Yet another hip?", *Data & Knowledge Engineering*, Vol. 41, No. 2-3, pp. 205-227.

ebXML, 2003, *Electronic Business XML*, http://www.ebxml.org

Economist, 2001, "Survey: Battle of the platforms", *Economist*, April 14, pp. S15-S19.

Fitzgerald, M., 2001, *Building B2B Applications with XML: A Resource Guide*. Wiley, New York, NY.

Fürst, K., Schmidt, T., 2001, "Turbulent markets need flexible supply chain communication", *Production Planning & Control*, Vol. 12, No. 5, pp. 525-533.

Goldfarb, C., Prescod, P., 2002, *Charles F. Goldfarb's XML Handbook*. Prentice-Hall, Upper Saddle River, NJ.

Glushko, R., Tenenbaum, J., Meltzer, B., 1999, "An XML framework for agent-based e-commerece", *Communication of the ACM*, Vol. 42, No. 3, pp 106-114.

Hasselbring, W., Weigand, H., 2001, "Languages for electronic business communication: State of the art", *Industrial Management & Data Systems*, Vol. 101, No. 5, pp. 207-216.

IBM, 2003, *webSphere Server*, http://www.ibm.com

ISO, 1986, *Standard Generalized Markup Language*, ISO 8879.

ISO, 1997, *Open-EDI Reference Model*, ISO/IEC 14662.

Jones, R., 2001, "B2B integration", *Manufacturing Engineer*, Vol. 80, No. 4, pp. 165-168.

Kalliokoski, P., Seilonen, I., Ollus, M., Koskinen, K., 2000, "Virtual enterprise co-operation model for web-enabled production management for SME-networks". In Stanford-Smith, B., Kidd, P. (Eds.), *E-Business: Key Issues, Applications and Technologies*, Vol. 2. e-2000, October 18-20, Madrid, Spain. IOS Press, Amsterdam, The Netherlands. Pp. 639-645.

Kay, M., 2001, *Saxon*, http://users.iclway.co.uk/mhkay/saxon

Koski, H., Rouvinen, P., Ylä-Anttila, P., 2001, *Uuden talouden loppu (The End of New Economy)*? ETLA B-184, SITRA 245, Taloustieto, Helsinki, Finland.

Kärkkäinen, S., Maunuksela-Malinen, P., Saloranta, A., 2001, *Yritysten välinen sähköinen liiketoiminta. EDI/OVT:n käyttö Suomessa (Business-to-Business E-Commerce. The Use of EDI in Finland)*. Julkaisusarjan osa 4, Finnish Information Society Development Centre, Helsinki, Finland.

Li, H., 2000, "XML and industrial standards for electronic commerce", *Knowledge and Information Systems*, Vol. 2, No. 4, pp. 487-497.

Linthicum, D., 2001, *B2B Application Integration: E-Business-Enable Your Enterprise*. Addison-Wesley, Boston, MA.

Lu, E., Tsai, R., Chou, S., 2001, "An empirical study of XML/EDI", *Journal of Systems and Software*, Vol. 58, pp. 271-279.

Luoma, J., Muhonen, T., Huomo, T., 1999, *Uudistuva tietotekniikka-arkkitehtuuri (Renewing Information Technology Architecture)*. HM&V Research, Espoo, Finland.

Microsoft, 2003, *BizTalk Server*, http://www.microsoft.com/biztalk

OAG, 2003, *Open Applications Group Integration Specification*, http://www.openapplications.org/oagis

OASIS, 2003, *Universal Business Language*, http://www.oasis-open.org/committees/ubl

OBI, 2003, *Open Buying on the Internet*, http://www.openbuy.org

Olson, E., Williams, P., 2001, *E-Commerce with Web-Based Standards: A Case Study in Implementing RosettaNet Process Standards at Avnet*. White paper, PriceWaterhouseCoopers LLP.

OMG, 2001, *Unified Modeling Language*, http://www.omg.org/uml

papiNet, 2002, *From Theory to Reality Using the papiNet XML Standards: A Case Study by StoraEnso North America*. White paper, papiNet Consortium.

papiNet, 2003, *papiNet*, http://www.papinet.org

Pawar, K., Driva, H., 2000, "Electronic trading in the supply chain: A holistic implementation framework", *Logistics Information Management*, Vol. 13, No. 1, pp 21-32.

Porter, M., 1985, *Competative Advantage: Creating and Sustaining Superior Performance*. Free Press, New York, NY.

Reimers, K., 2001, “Standardizing the new e-business platform: Learning from EDI experience”, *Electronic Markets*, Vol. 11, No. 4, pp. 231-237.

RosettaNet, 2001, *Case Study: Arrow and UTEC Replace EDI-Based Purchase Order Process with RosettaNet Standards*. White paper, RosettaNet Consortium.

RosettaNet, 2002a, *Achieving Streamlined Operations through Collaborative Forecasting and Inventory Management: STMicroelectronics E-Chain Optimization Project*. White paper, RosettaNet Consortium.

RosettaNet, 2002b, *Measuring Business Benefits of RosettaNet Standards: A Co-adoption Model Conducted by the University of Illinois*. White paper, RosettaNet Consortium.

RosettaNet, 2003, *RosettaNet*, http://www.rosettanet.org

Russell, S., Norvig, P., 1995, *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ.

SAX, 2002, *Simple API for XML*, http://www.saxproject.org

Shim, S., Pendyala, V., Sundaram, M., Gao, J., 2000, “Business-to-business e-commerce frameworks”, *IEEE Computer*, Vol. 33, No. 10, pp. 40-47.

SoftwareAG, 2003, *EntireX*, http://www.softwareag.com

Staab, S., Aalst, W. van der, Benjamins, V., Sheth, A., Miller, J., Bussler, C., Maedche, A., Fensel, D., Gannon, D., 2003, “Web services: Been there, done that?”, *IEEE Intelligent Systems*, Vol. 18, No. 1, pp. 72-85.

Statistics Denmark, Statistics Finland, Statistics Norway, Statistics Sweden, 2001, *Use of IC in Nordic Enterprises 2000/2001*. Statistics Norway, Kongsvinger, Norway.

Supply Chain Council, 2003, *Supply Chain Operations Reference*, http://www.supply-chain.org

Sun, 2000a, *Java 2 Platform Standard Edition*, http://java.sun.com/j2se

Sun, 2000b, *Java API for XML Processing*, http://java.sun.com/xml/jaxp

Sundaram, M., Shim, S., 2001, “Infrastructure for B2B exchanges with RosettaNet”. In *The 3rd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*. WECWIS 2001, June 21-22, Milpitas, CA. Pp. 110-119.

Tan, K., 2001, “A framework of supply chain management literature”, *European Journal of Purchasing & Supply Management*, Vol. 7, No. 1, pp. 39-48.

Tibco, 2003, *ActiveEnterprise*, http://www.tibco.com

UDDI, 2002, *Universal Description, Discovery and Integration*, http://www.uddi.org

UNECE, 2003, *EDIFACT*, http://www.unedifact.org

W3C, 1999a, *Extensible Stylesheet Language Transformations*, http://www.w3.org/TR/xslt

W3C, 1999b, *Hypertext Markup Language*, http://www.w3.org/MarkUp

W3C, 1999c, *Resource Description Framework*, http://www.w3.org/RDF

W3C, 2000a, *Extensible Markup Language*, http://www.w3.org/TR/REC-xml

W3C, 2000b, *Simple Object Access Protocol*, http://www.w3.org/TR/SOAP

W3C, 2001a, *Web Services Description Language*, http://www.w3.org/TR/wsdl

W3C, 2001b, *XML Schema Definition Language*, http://www.w3.org/XML/Schema

W3C, 2002, *Document Object Model*, http://www.w3.org/DOM

W3C, 2003a, *RDF Schema*, http://www.w3.org/TR/rdf-schema

W3C, 2003b, *Web Ontology Language*, http://www.w3.org/TR/owl-ref

webMethods, 2003, *Integration Platform*, http://www.webmethods.com

Westarp, F. von, Weitzel, T., Buxmann, P., König, W., 1999, "The status quo and the future of EDI: Results of an empirical study". In Pries-Heje, J., Ciborra, C., Kautz, K., Christiaanse, E., Avison, D., Heje, C. (Eds.), *Proceedings of the 7th European Conference on Information Systems*. ECIS 99, June 23-25, Copenhagen, Denmark. Copenhagen Business School, Copenhagen, Denmark. Pp. 719-731.

WfMC, 2003, *XML Process Description Language*, http://www.wfmc.org

Willcocks, L., Lester, S., 1999a, "Information technology: Transformer or sink hole?" In Willcocks, L., Lester, S. (Eds.), *Beyond the IT Productivity Paradox*. Wiley, Chichester, United Kingdom. Pp. 1-36.

Willcocks, L., Lester, S. 1999b, "In search of information technology productivity: Assessment issues". In Willcocks, L., Lester, S. (Eds.), *Beyond the IT Productivity Paradox*. Wiley, Chichester, United Kingdom. Pp. 69-97.

xCBL, 2003, *XML Common Business Library*, http://www.xcbl.org

XML/EDI Group, 2001, *XML/EDI*, http://www.xmledi-group.org

Zhao Y., Sandahl, K., 2000, "XML-based frameworks for Internet commerce". In *Proceedings of the 2nd International Conference on Enterprise Information Systems*. ICEIS 2000, July 5-7, Stafford, United Kingdom. Pp. 511-516.

Zuboff, S., 1988, *In the Age of the Smart Machine: The Future of Work and Power*. Basic Books, New York, NY.

Appendix

Appendix 1: XML in the BNF

The Backus-Naur form is a formal notation used to describe syntax of languages (Aho et al. 1986). The following grammar expresses a language of the XML Standard 1.0 without DTDs and namespaces. Figure A1 shows those productions of this grammar that contain non-terminal symbols, e.g. Letter, on the right-hand side of the production. Terminal symbols are enclosed in quotes, e.g. “<”.

```
Document ::= Element
Element ::= EmptyTag | StartTag Content EndTag
EmptyTag ::= “<” Name (S Attribute)* S? “/>”
Name ::= (Letter | “_” | “:”) (NameChar)*
Letter ::= BaseChar | Ideographic
NameChar ::= Letter | Digit | “.” | “-” | “_” | “:” | CombiningChar | Extender
Attribute ::= Name S? “=” S? Value
Value ::= “'” ([^<&"] | Reference)* “'” | “"” ([^<&'] | Reference)* “"”
Reference ::= “&” Name “;” | CharRef
StartTag ::= “<” Name (S Attribute)* S? “>”
Content ::= CharData? ((Element | Reference | CDSect | PI | Comment) CharData?)*
CDSect ::= “<![CDATA[” (Char* - (Char* “]]>” Char*)) “]]>”
PI ::= “<?” Name - ((“X” | “x”) (“M” | “m”) (“L” | “l”)) (S (Char* - (Char* “?>” Char*)))? “?>”
Comment ::= “<!--” ((Char - “-”) | (“-” (Char - “-”)))* “-->”
EndTag ::= “</” Name S? “>”
```

Figure A1: Productions of the grammar

This grammar also has ten other productions that consist of terminal symbols on the right-hand side of the production. If all the productions of the grammar have a single non-terminal symbol on the left-hand side of the production, the language is context-free (Russell and Norvig 1995). Since XML satisfies this requirement, it is a context-free language.

Appendix 2: Java interface of the processors

The engine and processors are not fitted together in compile-time but the engine loads and calls the processors in run-time. Figure A2 shows a CAProcessor that implements this interface.

```
public interface CAProcessor {
  public void setInput(Integer nro, Object val);
  public Exception execute();
  public Object getOutput();
}
```

Figure A2: CAProcessor

Each processor has a constructor with at most one argument of the String type. This argument is often an URI to the configuration data. The “setupInput” method sets an object “val” as an input in a variable that has a running number “nro” (0, 1, …). It is possible that the processor has no inputs. The “execute” method performs the process with the given inputs. This method returns an exception if it has occurred. Otherwise, it returns null. Finally, the output of the processor is obtained by the “getOutput” method. If the process has no output, this method returns null.

Appendix 3: A DTD for system configuration

The engine has one file that contains its system configuration data. This data includes attributes “dtd”, which is a URI to a DTD for an interaction request (Appendix 9), and “html”, which is a URI to an XSLT translation to transform an interaction response into the HTML format. An interaction is identified by a unique triple of attributes “operation”, “content”, and “document”. A content of the interaction is a URI to an interaction definition. Figure A3 presents a DTD for system configuration.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE configuration [
<!ELEMENT configuration (interaction)*>
<!ATTLIST configuration
  dtd CDATA #REQUIRED
  html CDATA #IMPLIED
>
<!ELEMENT interaction (#PCDATA)>
<!ATTLIST interaction
  operation (query|receive|send) "query"
  format (vsf|xcbl) "vsf"
  document (demand-forecast|invoice|order|order-list|order-response) "order"
>
]>
```

Figure A3: A DTD for system configuration

Appendix 4: A DTD for interaction definition

An interaction definition has at least one operation. It may also have parameters and a result. Each parameter has a unique attribute “name”, attributes “type”, and “state”. This type is Date, Double Integer or String to construct an object with the parameter value, or a URI to a DTD to validate the parameter value. If the state has a value “variable”, the engine gives the parameter value. Otherwise, the parameter value is a content of the parameter defined in the interaction definition. Each operation has attributes “name” and “processor”. This name is unique so that parameters and other operations do not have the same name. The processor is a URI to a class to construct a processor object. If the operation has an attribute “configuration”, which is a URI to its configuration data, the processor object is constructed with this data. The operations may also have inputs. The input has an attribute “name” that refers to the name of the parameter or previous operation. If there is a result, its value is given to the engine. The result has a name that refers to the name of the operation. Figure A4 presents a DTD for an interaction definition.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE interaction-definition [
<!ELEMENT interaction-definition (parameter*,operation+,result?)>
<!ELEMENT parameter EMPTY>
<!ATTLIST parameter
  name CDATA #REQUIRED
  type CDATA #REQUIRED
  state (constant|variable) "variable"
>
<!ELEMENT operation (input)*>
<!ATTLIST operation
  name CDATA #REQUIRED
  processor CDATA #REQUIRED
  configuration CDATA #IMPLIED
>
<!ELEMENT input EMPTY>
<!ATTLIST input
  name CDATA #REQUIRED
>
<!ELEMENT result EMPTY>
<!ATTLIST result
  name CDATA #REQUIRED
>
]>
```

Figure A4: A DTD for interaction definitions

Appendix 5: Configuration of the Query Processor

Configuration data of the Query Processor contains an attribute “connection” to open a database, and a SELECT statement to create a result set in XML format from the data retrieved from this database. The SELECT statement may include a number of question marks (?). For each question mark, there must be an input. Figure A5 presents an example.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE query [
<!ELEMENT database-query EMPTY >
<!ATTLIST database-query
  connection CDATA #REQUIRED
>
]>
<database-query connection="jdbc:odbc:vsfdb">
  SELECT * FROM Orders WHERE order_id = ? and company _id= ?
</database-query>
```

Figure A5: Configuration data of the Query Processor

Since the Query Processor has a simple configuration, its configuration data is processed without the XML parser, and not validated against a DTD.

Appendix 6: Configuration of the Update Processor

Configuration data of the Update Processor contains an attribute “connection” to open a database. In order to manipulate the content of this database, the Update Processor has one input that may consist of a number of UPDATE, INSERT, and DELETE statements. Figure A6 presents an example.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE query [
<!ELEMENT database-update EMPTY >
<!ATTLIST database-update
  connection CDATA #REQUIRED
>
]>
<database-update connection="jdbc:odbc:vsfdb"/>
```

Figure A6: Configuration data of the Update Processor

The Update Processor also has a simple configuration. Therefore, its configuration data is not validated against a DTD, and no XML parser is used.

Appendix 7: Configuration of the Messenger

Configuration data of the Messenger contains an attribute “mail” that is a mail server for sending e-mails, and pairs of attributes “name” and “address”. If the first input of the Messenger equals with a name, the second input is sent to an address related to this name. The address is a URI, which has a prefix “mailto:” to send an e-mail, “http://” to send a message, and “file:” to save a file. Figure A7 presents an example with a DTD.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE partners [
<!ELEMENT partners (send)*>
<!ATTLIST partners
  mail CDATA #REQUIRED
>
<!ELEMENT send EMPTY>
<!ATTLIST send
  name ID #REQUIRED
  address CDATA #REQUIRED
>
]>
<partners mail="vttmail.vtt.fi">
  <send name="smtp" address="mailto:autjmn@vtt.fi"/>
  <send name="http" address="http://autjmn:8080/vsf/servlet/CA"/>
  <send name="file" address="file:vsf/CA.txt"/>
</partners>
```

Figure A7: Configuration data of the Messenger

Appendix 8: Configuration of the Access Processor

Configuration data of the Access Processor contains triples of attributes “user”, “password”, and “id”. If the first input of the Access Processor equals with a user and the second input with a password of this user, the Access Processor returns a value of the attribute id related to the user. Figure A8 presents an example with a DTD.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE permissions [
<!ELEMENT permissions (access)*>
<!ELEMENT access EMPTY>
<!ATTLIST access
  user ID #REQUIRED
  password CDATA #REQUIRED
  id CDATA #REQUIRED
>
]>
<permissions>
  <access user="jmn" password="aut" id="%"/>
<permissions>
```

Figure A8: Configuration data of the Access Processor

Appendix 9: A DTD for interaction request

A DTD for interaction request includes attributes “operation”, “format”, and “document” to identify the interaction to be performed. An attribute “presentation” tells the format of an interaction response. The DTD also contains parameter attributes whose values are conveyed to the interaction. These values can be XML documents included in the interaction request. Figure A9 presents a DTD for an interaction definition.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE interaction-request [
<!ELEMENT interaction-request (parameter)*>
<!ATTLIST interaction-request
  operation (query|receive|send) "query"
  format (vsf|xcbl) "vsf"
  document (demand-forecast|invoice|order|order-list|order-response) "order"
  presentation (html|plain|xml) "plain"
>
<!ELEMENT parameter (#PCDATA)>
<!ATTLIST parameter
  name CDATA #REQUIRED>
>
]>
```

Figure A9: A DTD for interaction requests

Appendix 10: An interaction request embedded in HTML

An interaction request can be embedded in a web page by a web form that is always sent using “post” method and “multipart/form-data” encoding. The attributes operation, format, document, and presentation are replaced by hidden inputs “interaction-request:operation”, “interaction-request:format”, “interaction-request:document”, and “interaction-request:presentation”. Since the interaction request is sent via the web page, the input “interaction-request:presentation” should always have a value “html”. Parameters of the interaction request “user”, “password”, and “order-id” are replaced by hidden inputs “parameter:user”, “parameter:password”, and “parameter:order-id”. Figure A10 presents an example with a DTD that corresponds to an example in Figure 13.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE>Query Purchase Order</TITLE>
  </HEAD>
  <BODY>
    <H1>Query Purchase Order</H1><P>
    <FORM METHOD="post" ACTION="http://autjmn:8080/vsf/servlet/CA"
    ENCTYPE="multipart/form-data">
      <INPUT TYPE="hidden" NAME="interaction-request:operation"
      VALUE="query">
      <INPUT TYPE="hidden" NAME="interaction-request:format"
      VALUE="xcbl">
      <INPUT TYPE="hidden" NAME="interaction-request:document"
      VALUE="order">
      <INPUT TYPE="hidden" NAME="interaction-request:presentation"
      VALUE="html">
  User:
      <INPUT TYPE="text" NAME="parameter:user" FRAMEWIDTH="4"
      SIZE="31" MAXLENGTH="256"><BR>
  Password:
      <INPUT TYPE="password" NAME="parameter:password" FRAMEWIDTH="4"
      SIZE="31" MAXLENGTH="256"><BR>
  Order ID:
      <INPUT TYPE="text" NAME="parameter:order-id" FRAMEWIDTH="4"
      SIZE="31" MAXLENGTH="256"><BR>
      <INPUT TYPE="submit" NAME="send" VALUE="Send">
    </FORM>
  </BODY>
</HTML>
```

Figure A10: A home page for an interaction request of the query purchase order