

ebXML (electronic business XML)

CURTIN
University of Technology
Department School of Computing

Postgraduate Diploma in Computer Science
Unit: Computer Science Project 552
Second Semester 2001

Project Supervisor: Dr. John Bui (buihh@cs.curtin.edu.au)

Project Student: Sacha Schlegel *
Student ID: 12253634
E-Mail: sach@schlegel.li

Document *Revision* : 1.4

Copyright (c) CURTIN University, Perth Australia and Sacha Schlegel.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.1
or any later version published by the Free Software Foundation;
with no Invariant Sections, with no Front-Cover Texts and
with no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU
Free Documentation License".

June 17, 2002

*<http://www.schlegel.li>

Abstract

Electronic business (exchange of business documents) in the area of inter-enterprise business is the next step in total electronic business. Dedicated business software systems are available in the areas of Enterprise Resource Planning, Customer Relationship Management, Electronic Procurement and Supply-Chain Management. The next step is to link the enterprise oriented software systems to a global electronic marketplace. ebXML stands for electronic business XML (eXtensible Markup Language. ebXML is a new specification worked out by OASIS and UN/CEFACT. ebXML provides a framework for a global electronic marketplace. The purpose of the study is to understand the publicly available ebXML specifications with particular interests in business processes and messaging. XML is used to define data structures. ebXML provides a promising new specification which is based on inter-enterprise business collaborations. Experiences from EDI (Electronic Document Interchange) helped in the process of the specification of the ebXML specification. The success of ebXML will be based on the prompt creation of re-usable components (defined common business processes, core components, business documents etc) stored at an ebXML Registry/Repository. Further ebXML has to be integrated in electronic business software systems. To allow Small to Medium Enterprises (SME) to participate in the global electronic marketplace . Free software / open source ebXML solutions can help the Small to Medium Enterprises (SME) to participate in the global electronic marketplace.

Contents

1	Introduction	6
2	Background	7
2.1	Electronic Business Introduction	7
2.2	Enterprise Resource Planning	7
2.3	Customer Relationship Management	8
2.4	Supply Chain Management	9
2.5	Electronic Procurement	9
2.6	Electronic Markets	10
2.7	Integration	11
2.8	Related technologies	12
2.8.1	EDI (Electronic Data Interchange)	12
2.8.2	XML (eXtensible Markup Language) as a data format	12
3	ebXML	15
3.1	ebXML by example	15
3.2	Introduction to ebXML	16
3.3	ebXML Registry/Repository	19
3.4	ebXML Business Processes	20
3.4.1	The ebXML Business Collaboration	22
3.4.2	Specification of a Business Transaction and its Business Document Flow	24
3.4.3	Specification of a Binary Collaboration	27
3.4.4	Specification of a Multiparty Collaboration	27
3.4.5	Specification of the Choreography	30
3.4.6	The ebXML Business Process Specification Schema UML Class Diagram	33
3.5	ebXML Collaboration Protocol Profile and ebXML Collaboration Protocol Agreement	33
3.5.1	Content of a CPP	36
3.5.2	Content of a CPA	39
3.6	ebXML Core Components	40
3.7	ebXML Messaging Service	40
3.7.1	Packaging Specification	41
3.7.2	ebXML SOAP Extensions	42
3.7.3	MSH Services	45
3.7.4	Reliable Messaging	45
3.7.5	Error Reporting and Handling	45
3.7.6	Security	45

4	Definition of Business Processes	47
4.1	Analysis Worksheets for modelling Business Processes	48
4.2	Common Business Processes	48
5	Conclusion	50
5.1	Summary	50
5.2	ebXML Problems	51
5.3	Possible future directions	52
A	APPENDIX List of the official ebXML Documents	53
B	APPENDIX Relating free software / open source Projects	53
B.1	openebXML - open ebXML Implementation	53
B.2	GNUe - GNU Enterprise	53
B.3	GNU Free Documentation License	56

List of Figures

1	e-Business Application Architecture	11
2	ebXML Overview	17
3	ebXML Business Collaboration	22
4	UML Diagram of a Business Transaction	25
5	UML Diagram of document flow	26
6	UML Diagram of a Binary Collaboration	28
7	UML Diagram of a Multiparty Collaboration	29
8	UML Diagram of a Choreography	31
9	The ebXML Business Process Specification Schema UML Class Diagram	34
10	Collaboration-Protocol Profile and Agreement Specification .	35
11	The ebXML Message Structure	41
12	Graphical representation of the Porter Value Chain	48

List of Tables

1	Current ebXML available documents from http://www.ebxml.org	54
---	--	----

1 Introduction

The world of business is old and has a significant impact on our lives in the western world. New theories have been introduced and new ways of doing business have evolved. With the beginning of the computer era business has quickly adopted the usage of new technologies to advance. Today companies and organisations are heavily supported by software systems to conduct business. Early software systems were mostly internal company focused solutions and did not encourage enough inter-enterprise collaborations. The exchange of business documents between enterprises became the focus and standards like EDI (Electronic Document Interchange) ¹ emerged. ebXML can be viewed as the next generation of EDI. ebXML provides a framework for a global electronic marketplace.

ebXML is a joint initiative by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT)² and the Organisation for the Advancement of Structured Information Standards (OASIS)³. This initiative specified ebXML and provides the ebXML specifications per May 2001, for download at <http://www.ebXML.org>. Leading software companies have started to implement the ebXML specifications for their software packages.

However there is a current gap of interest in the computer science academic world in what the current real world business activities and specifications of ebXML are.

The objective of this report is to give background information on electronic business in general and to introduce the ebXML specifications. In particular the ebXML Business Process Specification Schema and the ebXML Messaging Service are shown.

Once the ebXML specifications are understood some ebXML related academic research can be done in order to solve some of the ebXML specific problems.

¹<http://www.unedifact.org>

²<http://www.uncefact.org>

³<http://www.oasis.org>

2 Background

This chapter provides backgrounds in electronic business as well as technologies.

2.1 Electronic Business Introduction

This chapter introduces electronic business and describes interesting aspects of it. Electronic business can be broken down into several parts, which are customer relationship management, enterprise resource planning systems, supply chain management, electronic procurement and electronic markets. These parts will be introduced in this chapter.

In general one of the most important aspects of electronic business is the complete integration of all parts of electronic business. All software systems have to communicate and integrate with all other software systems. There is no point in having island solutions. Pressure and competition in the market demand a fully integrated company.

If we look at simple desktop software suites we can imagine how complex these systems will become. Today in 2001 we have seamless integration within one suite but when we bring down the borders and explore interoperability between different components from different suites, maybe different operating systems, different programming languages, different hard- and software architectures and different database systems, we become aware of all the problems with compatibility, integration, security and many other aspects.

A general idea is to use new technology to support the daily business of a company, organisation or non-profit institute. Business is quite old and we see common business processes in various different industries (e.g. banking, insurance and chemical). Technology is used to empower employees, to automate common processes, to get closer to the customer and save costs, save expensive labour and time to market as well as the general competition pressure of the market. Today companies are doing electronic business but are missing the big picture of collaboration. Companies should not only concentrate on its customers and its processes but enable the seamless cooperation with other companies.

2.2 Enterprise Resource Planning

ERP systems can be seen as the backbone of electronic business [15]. Today big companies run ERP systems to stay in business. The days with folders, copy machines, immense paper work should be gone and all the work of a company should run by an ERP system. Companies in the world have similar activities within the company. Finance, Human Resource, Logistics, Planning, Manufacturing are some typical activities in almost every com-

pany. In human resources for example a company has employees and the company has to pay them. The company has to keep track of the addresses, holidays, overtime, social payments (insurance, retirement), payment itself, bonus systems, government taxes and other information. The latest human resource software systems help even to analyse fluctuations and to analyse employee satisfaction.

With ERP, the processes of companies are modelled in a computer system but still with the ability to change and to extend. Processes are central in ERP systems. First the company is analysed and common business processes have to be found. Those processes get defined in the ERP system. Very often though a company changes its practice in order to “simply” adjust for a new ERP system.

Today, companies need ERP systems. Companies need information in real time. A chief finance officer (CFO) has to know the current financial situation in detail at any time, anywhere. Decision Support Systems support employees to make decisions.

2.3 Customer Relationship Management

New business theory tells us that the customer has the highest priority in any company. If there are no customers there is no company. The company has to be build around the customer. Everything has to be done to fullfill the needs and wishes of the customer. The customer is in the centre and not the company itself [15].

Customer Relationship Management itself has different parts like, service, marketing, and sales.

Customer data has to be central. All information about the customer has to be stored in only one place. Every person within the company has to have access to an up-to-date real time customer database. The customer information must be 100% complete. The customer has to be able to access the company through any channels (and vice versa) like telephone, personal contact, fax, web, e-mail, mobile services.

It is important that the information about the customer is consistent. Some examples help to visualise the needs for customer relationship management systems.

- The customer calls the call centre. The person in the call centre has up-to-date information about the customer. e.g the employee sees, that the customer called yesterday about a problem. The call centre person could greet the customer by name and ask if he could solve the problem with the product.
- The person in the call centre sees that the customer has product number X and makes the customer aware that there is a free upgrade if he/she wishes to get it by mail.

- The employee interacting with the customer sees that the customer has product Y and sees that the company has a new product which supercedes product Y and asks the customer if he/she wants to get some more information on the new product.
- The customer logs to the web site and gets his personalised web-interface. On the interface there might be important information about updates of products or some special offers tailored for this individual customer according its individual composition of products.

The main message of CRM is: Its all about your customers.

2.4 Supply Chain Management

This is about supplier and consumer. A company which produces goods needs parts from suppliers. Supply chain management helps both the supplier and the consumer. The ERP system of the consumer can communicate with the ERP system of the supplier. In this case the consumer can check the inventory which in fact is not his inventory but the suppliers inventory. Of course both parties must agree on how this is done.

With a good SCM inventory, costs can be lowered. Kotok and Webber [16] list a project called Collaborative Planning, Forecasting and Replenishment (CPFR) project. According to that project companies “share” more data and as a result they reduce expensive inventories and costs. The supplier company as an example has 10 big customers. With its customer data the supplier can optimise its business dramatically. The customers share their planning data, their forecast data, maybe even their planned specials (which means a temporary higher demand). With this data the supplier can optimise its inventory. The supplier can plan and forecast better. The supplier knows what, when and how much its customers need. So the supplier can provide the goods on time, thanks to the information of its customers. The customers can expect to have the goods delivered on time. The supply chain can spread across multiple hops in the value-added business chain.

2.5 Electronic Procurement

Most companies need everyday material to work. These products/materials are not necessary for a product directly. Today most companies know exactly how much a produced good costs; the fixed costs plus the variable costs. These companies often do not know how much they spent for small parts like rulers, pens, paper, printers, computers, and chairs.

The process to get these materials is expensive for all companies. Often a person in a department has to order some material, he/she has to fill out a paper and go to the boss of the department. The boss of the department signs the paper and then the request goes to a company-wide order office.

This order office waits a couple of days to get a certain quantity of the same items to order and get a discount in return. When the goods arrive the company has to distribute the materials and pay the bill. This takes a long time and too many person are involved.

E-Procurement has a different approach: Empower the employee. First the company typically gets one, two or more catalogues of products. The E-Procurement system sets rules for the ordering of products like the maximum value per order without approval, the number of orders exclusion of some products for certain employees (or groups) etc. E-Procurement uses workflow technology. A rule might say an item with a price higher than X, the department boss has to sign the order. The employee orders the item and this triggers a signing request at the department's boss' desktop. The boss signs the order by simply accepting it. The E-Procurement system then sends the order message to the companywide procurement server which directly sends the order in a defined way (maybe ebXML compliant) to the supplier of the item. The supplier makes sure that the company gets the latest catalogue with the latest prices and conditions. With this electronic process an order of any item can be a matter of minutes until the order is in the supplier electronic system [16].

2.6 Electronic Markets

Electronic markets are platforms where buyers and sellers (and maybe a market maker) can meet. Actually this can be a computer or a farm of computers. The market has to provide the infrastructure so that companies can access the market. Today some companies (SAP ⁴, CommerceOne ⁵, Oracle ⁶ and more) are providing electronic marketplaces.

According to Choudhury [3] an electronic market has three main market making functions:

- The *Identification*
is about the possibility for the user to search for a product. It depends on the complexity to describe the product as well as if the product is low in asset specification. The user gets a list of suppliers
- The *Selection*.
If the user has access to the price of the items, he/she can compare prices among suppliers. This assumes that the supplier publishes prices. Another way to get a price is by running an auction.
- The *Execution*
is the deal. If a supplier and buyer get matched, information is ex-

⁴<http://www.sap.com>

⁵<http://www.commerceone.com>

⁶<http://www.oracle.com>

changed (shipping address, payment possibility, etc.) to proceed with the transaction.

Obviously electronic markets are open 24 hours, 7 days a week.

2.7 Integration

The most difficult part is to integrate the different parts of electronic business. There will be parts of e-business from different vendors and companies have to integrate new applications into their landscape of legacy applications.

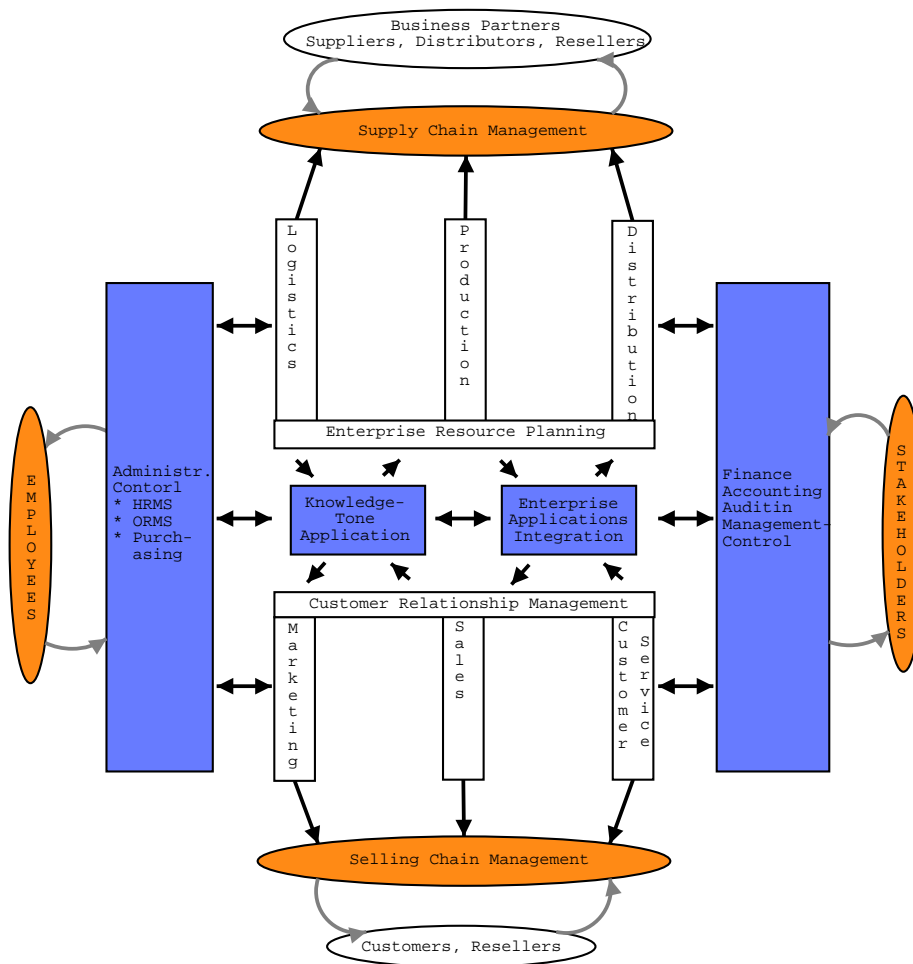


Figure 1: e-Business Application Architecture (Adapted from Kotok and Webber, 2001)

Figure 1 shows the diagram how the individual e-business parts are related to each other and interact with each other.

2.8 Related technologies

Background information about common technologies in the electronic business environment and specifically ebXML related technologies is available. There is great information available on the Internet of the following sub chapters: EDI and XML.

2.8.1 EDI (Electronic Data Interchange)

After mastering the communication between computer systems in the very early days of computing the interchange of business data becomes important. EDI was initiated in the 1960's to address the foundation of interchanging business data between computer systems. EDI exchanges structured business data in an agreed standard format by the trading partners. EDI established two major standards: ANSI X.12 and UN/EDIFACT (published as an International Standard, ISO 9735). The ANSI X.12 standard is used in North America and UN/EDIFACT globally. Industry specific EDI standards exist and SWIFT (Society for Worldwide Interbank Financial Transactions) is a well known EDI standard in the banking industry. There are standards for all major industries available today ([1], [2], [17]).

2.8.2 XML (eXtensible Markup Language) as a data format

XML (eXtensible Markup Language) is a standard from the world wide web consortium ⁷ and still evolving. There are many introductions to XML and many books are written about XML. XML in 10 points is an XML introduction which is used here and can also be read at <http://www.w3.org/XML/1999/XML-in-10-points>

1. XML is for structuring data

XML has basically two parts: the definition of a structure and one or many instances of it. First the structure is defined. An address for example has typically one street name, one street address, one country, one zip code, one area code etc. An instance address which conforms to the structure is created next. XML Schema provides data type definition and inheritance.

2. XML looks a bit like HTML

XML also uses tags like in HTML. The advantage is, that the tags can be defined. A tag can have attributes. A tag typically has attributes and content and/or sub tags (called child elements).

3. XML is text, but isn't meant to be read

It is very convenient to have XML human readable. But XML is

⁷<http://www.w3c.org>

mainly used as application to application data format. At both ends (application) XML parsers parse the XML document for further processing its content.

4. XML is verbose by design
As mentioned above, XML is text based. The XML document can be read. To transfer the XML document (file) it can be compressed to save bandwidth.
5. XML is a family of technologies
There are many further technologies in connection with XML. XSLT adds a style-sheet with rules to an XML document. With a XSLT style-sheet XML can be converted to HTML by simple rules. XPointer can point to parts within an XML document. XLink is used to link/reference other XML documents.
6. XML is new, but not that new
XML is a subset of SGML which was developed in the early '80s. SGML is too complicated for general usage and XML provides a good subset.
7. XML leads HTML to XHTML
HTML is a subset of XML or a defined vocabulary with elements like `< p >` for paragraph etc. XHTML is XML-based but has more or less the same elements as general HTML.
8. XML is modular
An XML element can have multiple child elements. And each child element can have further child elements. When an XML document uses more than one XML definition it might happen, that there are elements with the same name. The solution to avoid name collision is to use a namespace mechanism.
9. XML is the basis for RDF and the Semantic Web
The Semantic Web Group wants to add semantic to the content of the web. A browser does not know what an address is. If the browser could understand that the content of the address element is a geographical address the browser/computer could understand the content of web sites. RDF (Resource Description Framework) will address the semantic meaning of web.
10. XML is license-free, platform-independent and well-supported
XML is an open standard. Because of its nature to define data structure it is platform and programming language independent. There are free available parsers in almost any programming language on almost any platform.

It is possible to visit the w3c.org web site for the specifications of XML and related XML technologies. XML technologies are used by ebXML. These are typically XML, XSLT, XLink, XPointer, XML namespaces, SOAP, and XML Encryption.

3 ebXML

ebXML (electronic business eXtensible Mark-up Language) is a joint initiative by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organisation for the Advancement of Structured Information Standards (OASIS). The ebXML initiative provides the specifications for doing electronic business based on XML technology. The specifications of the initiative reached version 1.0 in May 2001 after 18 months of work. All documents (specifications, reports and white papers) can be downloaded from the web site of ebXML at <http://www.ebxml.org>. Further there are active mailing lists and archives of those mailing lists available at their site.

This section starts with an ebXML example. A short introduction to ebXML including a general example follows. Further the main parts of ebXML are mentioned. The main parts of ebXML are the Registry/Repository, Business Processes, Collaboration Protocol Profile and Collaboration Protocol Agreement, Core Components and Messaging.

3.1 ebXML by example

A small company in Brazil (around 20 employees) is in the imaging business. They have a few very talented artists who are doing contemporary South American art. The company has a good network of contractors, in case they need more artists. The company is very popular in Brazil because of their very good work. The company is mainly in the local national print media market present. The company wants to establish a good international reputation. The company tries to achieve this with a web site with and contact addresses and some presentations of their previous art work.

Another small company in Perth (around 40 employees) is involved in web projects. This Perth based company does a lot of web site integration to back-end software systems and is technically excellent but does not have the appropriate art expertise. The company is a well know web company within Australia and has many national customers. It happens, that the Perth based company gets a new project and is desperately looking for South American art to support a specific customer need.

Both companies hear of ebXML and have their business ready for ebXML. Thanks to ebXML, which is intended for Small to Medium Companies (SME), they can easily become part of the global electronic marketplace.

The Perth based company checks the ebXML Registry for a company who is into art (geographical location does not matter) and finds the company in Brazil. A contract is negotiated and both companies agree to do business together. The Brazilian company gets one or two contractors for this new project. After a couple of days the Brazilian company sends the first draft images back to Perth and the business in Perth is happy with

the images. The final images are transferred and in return the Perth based company transfers the money to Brazil. After success with the very expressive images from South America the business in Perth is so happy that they keep working together with the Brazilian company.

ebXML enabled both small businesses to meet and to do electronic business together. In reality it will be more likely to have bigger companies who have a big turnover on mass products to be involved in ebXML. But new businesses ideas and the usage of Internet with the latest technologies will create new business models.

3.2 Introduction to ebXML

The goal of ebXML is to provide a framework to enable the vision of a global electronic marketplace. ebXML relies on business processes. ebXML provides the framework which allows companies to register and to discover potential trading partners. Companies describe their business processes in a defined manner and register the resulting XML documents at the registry. The business processes display the public interface of the given company. The ebXML framework provides a mechanism to match companies with the same business processes. Once two companies are matched they know already the business processes they can follow. As the name implies the message flow between the ebXML registry and any companies which are XML based (see 2.8.2 for more info on XML). Further are business processes, business documents, business partners, core components etc all implemented in XML.

An important goal of ebXML is to enable Small to Medium Enterprises (SME) to do electronic business and to be part of the global electronic marketplace.

Picture 2 shows ebXML in action. The example is taken from the technical architecture specification [14].

The example shows how two companies involved in ebXML find each other and start electronic business.

1. Company A finds out about the electronic business opportunities based on ebXML. Company A browses the registry to see what is already available in the registry. Among the most important information available will be the industry wide common business processes. ebXML promotes the ideas of reusing predefined business processes. Company A will most likely find their core business processes defined in the repository. In the case of missing common business processes the company can define new ebXML business processes according to the ebXML Business Process Specification Schema [7] and the Business Process Analysis Worksheets & Guidelines [5]. Section 3.4 goes into more detail of the ebXML business process aspects. The best case

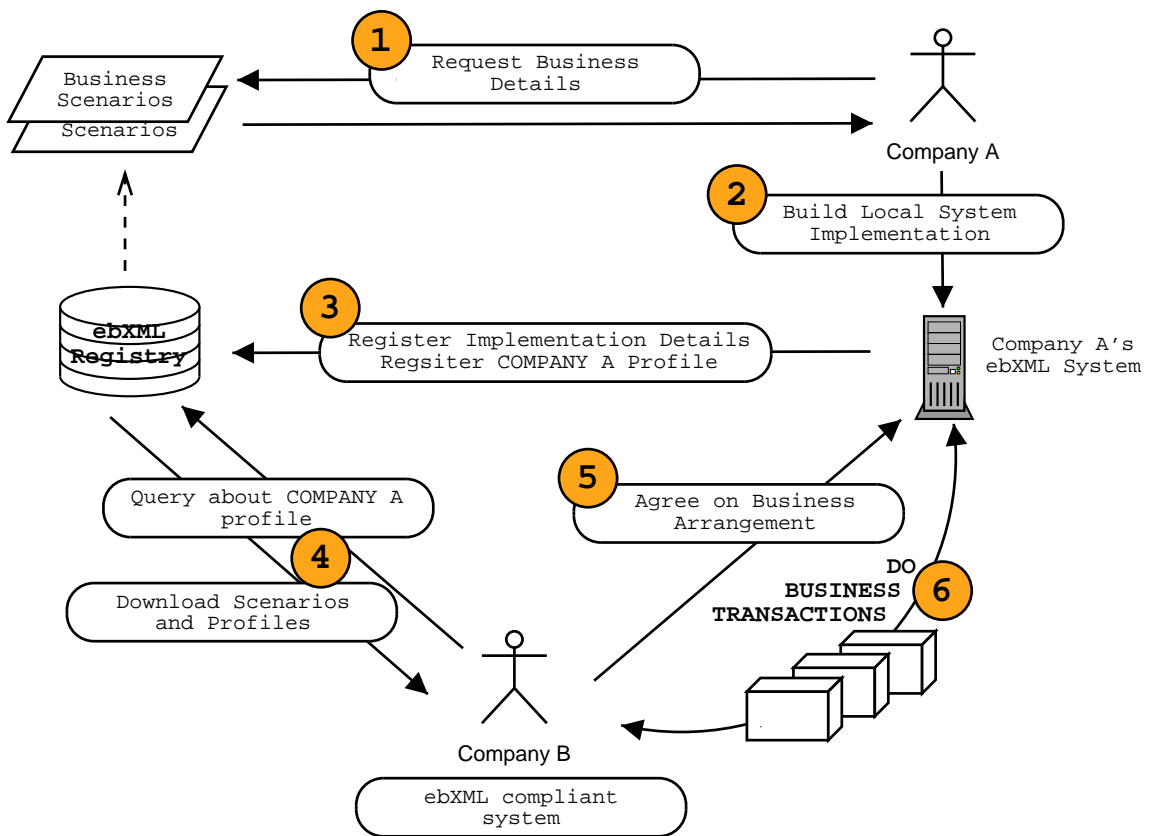


Figure 2: ebXML Overview (Adapted from the ebXML Technical Architecture specification)

for company A would be, if all the business processes of its industry are already stored in the ebXML Registry/Repository and company A just reuses the necessary business processes.

2. So company A decides to do electronic business the ebXML way and considers implementing a local ebXML compliant application called the Business Service Interface (BSI). In this case the company has to define itself in an ebXML compliant way. This means the company has to create a Collaboration Protocol Profile (CPP, the business profile). Basically a document which represents the supported business process capabilities, constraints and technical ebXML information of company A.
3. Company A submits its CPP to the registry. From that point on, company A is publicly listed in the repository and is likely to be discovered by other companies looking for new trading partners.
4. Company B is looking for new trading partners. Company B is already registered at the ebXML registry/repository. It starts a new query and receives the CPP of company A. Company B has then two CPP's: Company A's CPP and company B's CPP. ebXML then derives a third document from the intersection of the two CPP's called Collaboration Protocol Agreement (CPA).
5. Company B contacts company A directly and sends the newly created CPA for acceptance to company A. Upon agreement of the CPA by company A both companies are ready for electronic business.
6. Company B and company A then do electronic business directly according to the CPA. This means that both companies follow the business processes defined in the CPA.

This short example shows some important individual parts of ebXML, which will be described in the further chapters.

In ebXML there are three different functional phases.

1. The implementation phase

As mentioned in the example the ebXML registry/repository has to be filled with common business processes. Industries, organisations and companies have to come together and to define (or convert from older business process projects like EDI) the common business processes. A business process has many different business documents and those documents have fields like date, place, name, street, quality, quantity etc. All those data has to be defined and converted to XML documents to be stored in the ebXML repository. When, the business process are

defined then companies can reuse those business processes. A bottom-up approach is possible as well as a top-down approach. At the bottom we have the smallest entities. Probably integers, booleans, strings and the kind. A level higher we already might have date, time, street, etc. The next level could be address which reuses street, zip, country etc. All the way up elements reference other elements and at the top we get the CPA's.

2. The discovery and retrieval phase

After the first phase the repository will have a number of companies and new companies will start to query the repository and most likely find matching CPP's.

3. The run time phase

After the second phase and the derivation of Collaboration Protocol Agreements the involved companies can finally engage in electronic business in a defined way according to the CPA. The idea of ebXML is that the Business Service Interface gets configured with a Collaboration Protocol Agreement.

The main parts of the ebXML architecture are the ebXML Registry/Repository, the ebXML Business Processes, the ebXML Collaboration Protocol Profiles and Collaboration Protocol Agreements, the ebXML Core Components and the ebXML Messaging Service. The following chapters are based on the specifications of ebXML and they are regarded as an introduction and overview of the different major parts of ebXML. The content might not be deep enough for implementors. It is highly suggested to study the specifications instead.

3.3 ebXML Registry/Repository

The ebXML Registry/Repository can be viewed as a database. The original idea has envisioned a distributed ebXML Registry/Repository. The Repository is where the data gets stored (the physical database). The Registry on the other hand is where users query, retrieve and upload data. The Registry is like the interface of the Repository. The Registry itself is connected to the Repository in the background. There are a lot of different functions of these two ebXML entities.

There are different objects which have to be stored in the Repository. Beginning with the smallest entities these objects are the Core Components, the aggregated Core Components up to the Business Processes, Business Process Information Meta Models, Business Documents, Collaboration Protocol Profiles, Collaboration Protocol Agreements and more (some of these parts are explained in later chapters). These objects do not have to be in XML.

The ebXML Registry Services Specification v1.0 [13] has all the details of the ebXML Registry/Repository. The ebXML Registry Information Model v1.0 specification [12] deals with how the data is stored in the Registry/Repository, how versioning is done and how data can be updated, extended and deleted.

The ebXML Registry architecture is based on a client-server architecture. The communication can be based on ebXML Messaging Service (see 3.7 for information on Messaging Service) or simply by HTTP. In case the client is using a common web browser the web server has to use the Registry Interface to communicate with the Registry itself. The ebXML Registry Services Specification defines Registry Interfaces to ensure, that Registry clients can communicate with Registry Servers. The first Interface is the Interface Registry Service which has only two methods to get a) an Object Manager and b) an Object Query Manager. The two important interfaces are the Object Manger and the Object Query Manager. The list of methods of these interfaces should allow the reader to get a picture of the usage of the interface.

- The Object Manager Interface has the following methods:
 - approveObjects()
 - deprecateObjects()
 - removeObjects()
 - submitOjbects()
 - addSlots()
 - removeSlots()

- The Object Query Manager Interface has the following methods:
 - getClassificationTree()
 - getClassifiedObjects()
 - getContent()
 - getRootClassificationNodes()
 - submitAdhocQuery()

It is worth mentioning, that an object can be either a Core Component, a Business Process or anything else.

3.4 ebXML Business Processes

The electronic business process of ebXML is based around the execution of business processes.

A business process is roughly a business scenario about what happens, who the trading partners are, what the roles of the trading partners are, which documents the trading partners exchange, in what order the documents are exchanged and what information is in the documents (structure of the documents). In ebXML a Business Collaboration is a realisation of a Business Process.

The ebXML Business Process Specification Schema document [7] is well written and provides a very good view of ebXML Business Processes.

As mentioned before, ebXML has three different functional phases. Over the long run there will be Business Processes stored in the ebXML Registries. To get there, Business Processes first have to be modelled. ebXML based its modelling method on the UN/CEFACT Modelling Methodology (UMM) which is defined in the N090R9.1 specification (please check at the web site of UN/CEFACT for further details of UMM). UMM is a modelling methodology for business processes which describes what a business process is and what it contains. ebXML Business Process Specification Schema is modelled in UML (Unified Modelling Language ⁸) and converted to XML (see chapter 2.8.2 for infos on XML). It is also important to mention, that the ebXML Business Process Specifications do not specify the structure of Business Documents. Business Documents are composed from re-usable Business Information Objects and Business Information Objects are composed of re-usable Core Components.

To get from Business Processes to two Trading Partners doing business, ebXML envisioned the following process:

1. Business Process and Information Modelling.
2. ebXML Business Process Specification.
3. ebXML Collaboration Protocol Profile and Collaboration Protocol Agreement.
4. Configuration of the Business Service Interface software (application at each trading partner which handles the ebXML documents) with the Collaboration Protocol Agreement.

Figure 3 shows the ebXML Business Collaboration. ebXML Business Process Specification Schema defined the general Business Collaboration. Instances of Business Collaborations have to be worked out by companies, organisations but the class diagram, with associations, inheritance is given here. The Business Process Specification Schema [7] defines a business collaboration as follows:

⁸<http://www.uml.org>

Two or more business partners participate in the business collaboration through roles. The roles interact with each other through Business Transactions. The business transactions are sequenced relative to each other in a Choreography. Each Business Transaction consists of one or two predefined Business document flows. A Business Transaction may be additionally supported by one or more Business Signals.

The next chapters show how ebXML defines the Business Processes as Business Collaborations. The modelling is done in UML and converted to XML.

3.4.1 The ebXML Business Collaboration

The ebXML Business Collaboration is the realisation of a business process. The realisation is modelled in UML and converted to XML. The main parts of a business collaboration are:

1. Business Collaboration

A Business Collaboration is a set of Business Transactions between business partners. Each trading partner plays one or more roles in the collaboration. There are two different Business Collaborations: The Binary Collaboration and the Multiparty Collaboration. In the Binary Collaboration there are only two trading partners involved. In

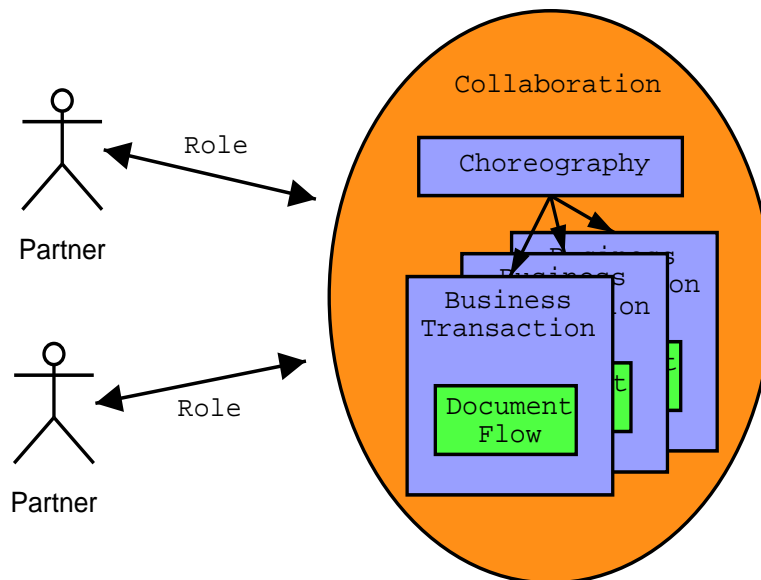


Figure 3: ebXML Business Collaboration (Adapted from the Business Process Specification Schema document)

a Multiparty Collaboration there are more than two trading partners involved. But a Multiparty Collaboration is synthesised from two or more Binary Collaborations.

A Binary Collaboration has several Business Activities. A Business Activity itself can be another Binary Collaboration (for recursion) or a concrete Business Transaction Activity.

2. Business Transaction

A Business Transaction Activity is basically a concrete Business Transaction. A Business Transaction takes place between *two* trading partners which play the opposite role, e.g. the “buyer” role and the “seller” role. A Business Transaction *always* has a requesting role and a responding role. It is important to mention, that a Business Transaction always fails or succeeds. The Business Transaction has Business Documents attached to it.

3. Business Document flows

As seen above, the Business Transaction has always two roles. The requesting role has one Business Document. The responding role may not have a Business Document e.g. a one-way notification has no responding Business Document. The structure of the Business Document itself is defined somewhere else. Of course ebXML reuses the Core Components (see later chapter for Core Components) or so called Business Information Objects from a Core Component Library to create the Business Documents.

4. Choreography

It is important to notice, that a Business Collaboration between two trading partners can have several Business Transactions (Business Activities). The Choreography defines the sequence of the Business Transactions. The Choreography is realised by a transition between two Business Transactions. A transition has guards associated with it. Choreographies are modelled by UML Activity Diagrams.

5. Patterns

Patterns are a set of predefined transaction interactions. Patterns are discovered commonalities. The ebXML E-Commerce Patterns document [10] explains more.

An example Business Process illustrates these parts: Company A provides a standardised catalogue to company B. Company B checks for the required product and asks company A if the required goods are deliverable. Depending on the answer, company B requests an offer and company A sends an offer for the requested quantity of product X to company B. Company B accepts the offer and sends an order back to company A. Company

A confirms the order and starts the shipment of the goods. Company A sends a shipment notification to company B including the bill. Company B gets the goods and pays the bill. This example scenario is easy and complex at the same time.

As noted above the Business Process Specifications are modelled with a UML class diagram (see figure 9 for the class diagram). In the class diagram we can see how the realisation of the main parts of a Business Collaboration are realised. A closer look at the class diagram helps understanding of how a Business Collaboration is built in detail.

3.4.2 Specification of a Business Transaction and its Business Document Flow

The Business Transaction is a central role within a Business Collaboration. Figure 4 shows the related UML class diagram. Basically a Business Activity uses a Business Transaction. A Business Transaction has *two* Business Activities which are the Requesting Business Activity and the Responding Business Activity. There is always a requesting activity and a responding activity. Both activities have common attributes (inherited from the abstract BusinessAction class) namely responsibility for the so called Business Signals, which are notification messages like: “Receipt Acknowledgement Signal” and “Acceptance Acknowledgement Signal”. A requesting activity has *only* 1 document envelope (whereas a document envelope class has zero to *n* documents, which reference Business Documents). The responding activity has zero or more document envelopes. That means, that a responding activity can have no documents at all.

The Business Process Schema Specification [7] includes the complete XML Schema code. Here are examples of XML instances to demonstrate a Business Transaction.

```
<BusinessTransaction name='''Notify of advanceshipment''>
  <RequestingBusinessActivity name='''>
    <DocumentEnvelope BusinessDocument name='''ASN''/>
  </RequestingBusinessActivity>
  <RespondingBusinessActivity name='''>
  </RespondingBusinessActivity>
</BusinessTransaction>
```

To define the flow of the Business Documents (which document follows which document within a business transaction) the specification uses an indirect definition. By default each requesting activity has a Business Document. Again, the structure of the Business Document is not defined in the Business Process Schema Specification. In the UML class diagram 5 the document envelope class *only* has a reference to the real document (the Business Document). Each requesting activity has a document but not the

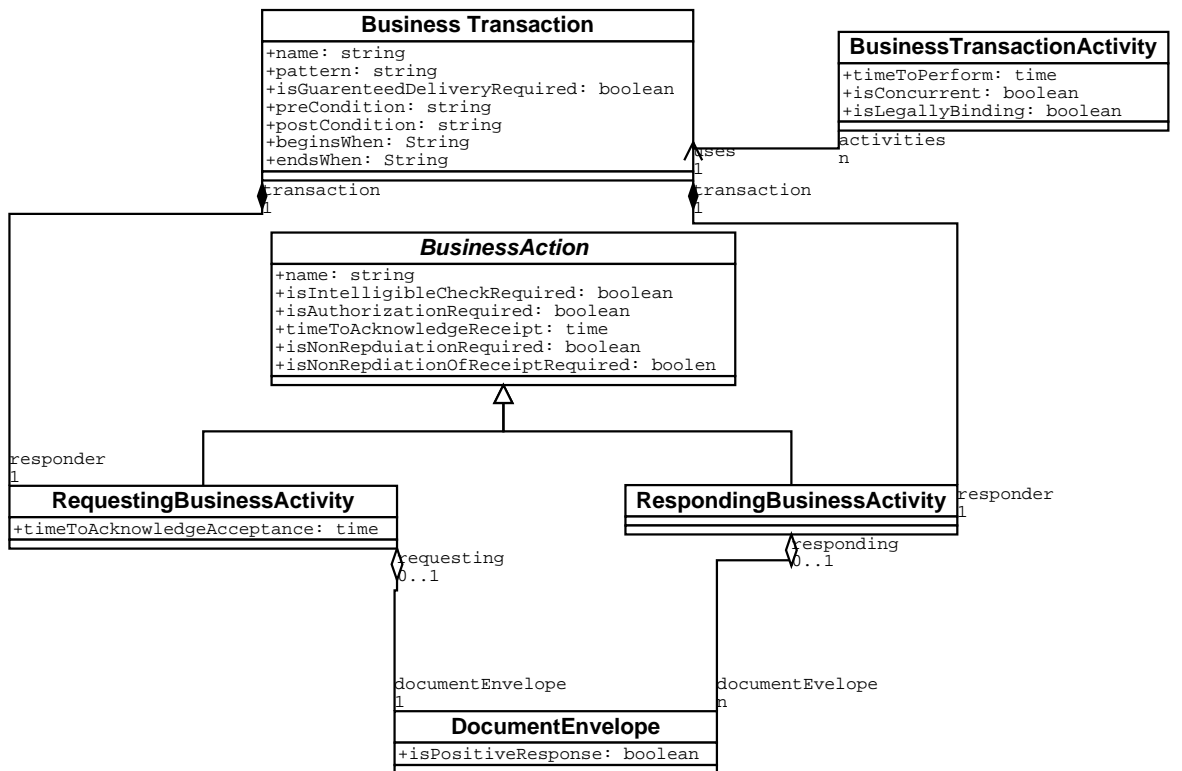


Figure 4: UML Diagram of a Business Transaction (Adapted from the Business Process Specification Schema document)

responding activity. A responding activity can have n documents (document envelopes). Further a document envelope can have n attachments for whatever is necessary. ebXML introduces the security of the documents. The Document Envelope and the Attachment class inherits from the DocSecurity class, which has attributes for encryption, signature and repudiation.

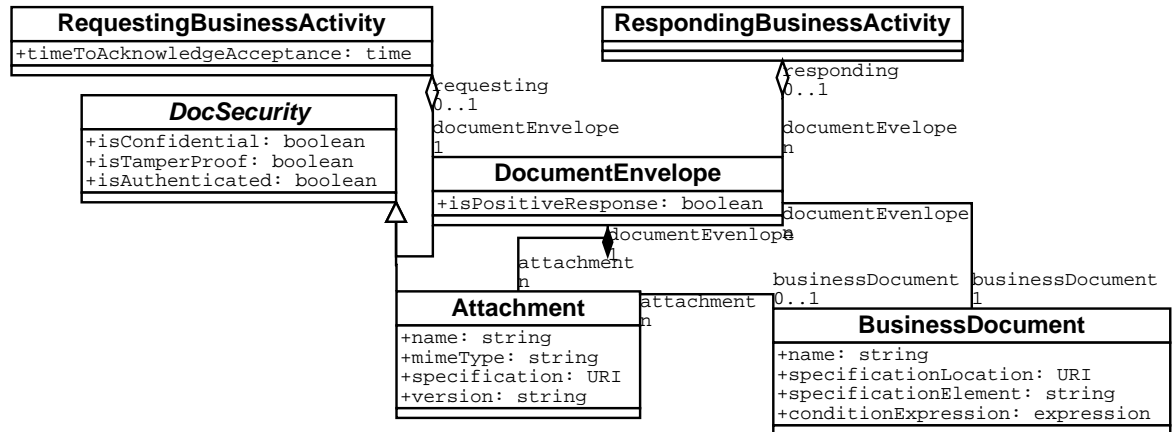


Figure 5: UML Diagram of document flow (Adapted from the Business Process Specification Schema document)

Again some XML source code will clarify how the Business Transaction uses Business Documents and how the flow of the documents is solved with the requesting activity and the responding activity.

```

<BusinessDocument name=''Purchase Order''
    specificationLocation=''somewhere''/>
<BusinessDocument name=''PO Acknowledgement''
    specificationLocation=''somewhere''/>
<BusinessDocument name=''PO Rejection''
    specificationLocation=''somewhere''/>
<BusinessDocument name=''Delivery Instructions''
    specificationLocation=''somewhere''/>

<Business Transaction name=''Create Order''>
  <RequestingBusinessActivity name=''''>
    <DocumentEnvelope isPositiveResponse=''true''
      BusinessDocument name=''ebXML1.0/PO Acknowledgement''/>
    <Attachment
      name=''DeliveryNotes''
      mimeType=''XML''
      BusinessDocument=''ebXML1.0/Delivery Instructions''
      specification=''''
      isConfidential=''true''
      isTamperProof=''true''
      isAuthenticated=''true''>
  
```

```

        </Attachment>
    </DocumentEnvelope>
</RequestingBusinessActivity>
<RespondingBusinessActivity name='''>
    <DocumentEnvelope BussinesDocument='''ebXML1.0/PO Acknowledgement''/>
    <DocumentEnvelope isPositiveResponse='''true''
        BussinesDocument='''ebXML1.0/PO Rejection''/>
</RespondingBusinessActivity>
</Business Transaction>

```

3.4.3 Specification of a Binary Collaboration

The next step is to build a Binary Collaboration from the defined Business Transactions. Again, the Binary Collaboration is the realisation of a Business Process and a Business Process can combine several Business Transactions. The class diagram from figure 6 shows, that in fact a Business Collaboration is realised by different states. Each Business Transaction Activity is a state. Further the Binary Collaboration has several authorised roles and each Business Activity (business state) comes from one authorised role and goes to an authorised role. As mentioned earlier a Business Activity can be another Binary Collaboration.

The example XML code shows the Binary Collaboration with the name “Firm Order”. The Binary Collaboration has one Business Transaction Activity with the name “Create Order” and two authorised roles, namely “buyer” and “seller”. The Business Transaction Activity references a Business Transaction “Create Order” defined earlier or somewhere else.

```

<BinaryCollaboration name='''Firm Order'' timeToPerform='''P2D''>
    <Documentation>
        timeToPerform = Period: 2 days from start of transaction
    </Documentation>
    <InitiatingRole name='''buyer''/>
    <RespondingRole name='''seller''/>
    <BusinessTransactionActivity name='''Create Order''
        businessTransaction='''Create Order''
        fromAuthorizedRole='''buyer''
        toAuthorizedRole='''seller''/>
</BinaryCollaboration>

```

3.4.4 Specification of a Multiparty Collaboration

A Multiparty Collaboration is used when more than two trading partners are involved in a a collaboration. For example a buying company, a selling company and a broker company. The Business Process Specification Schema allows multi party collaboration. Figure 7 shows how the Business Process Specification Schema allows Multiparty Collaboration.

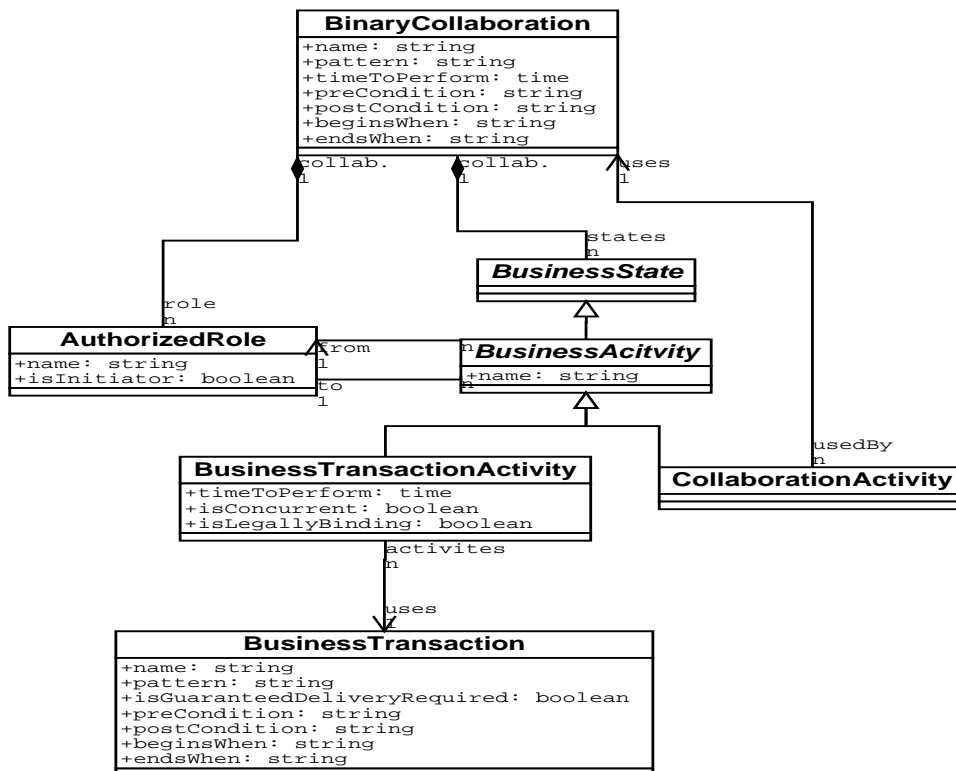


Figure 6: UML Diagram of a Binary Collaboration (Adapted from the Business Process Specification Schema document)

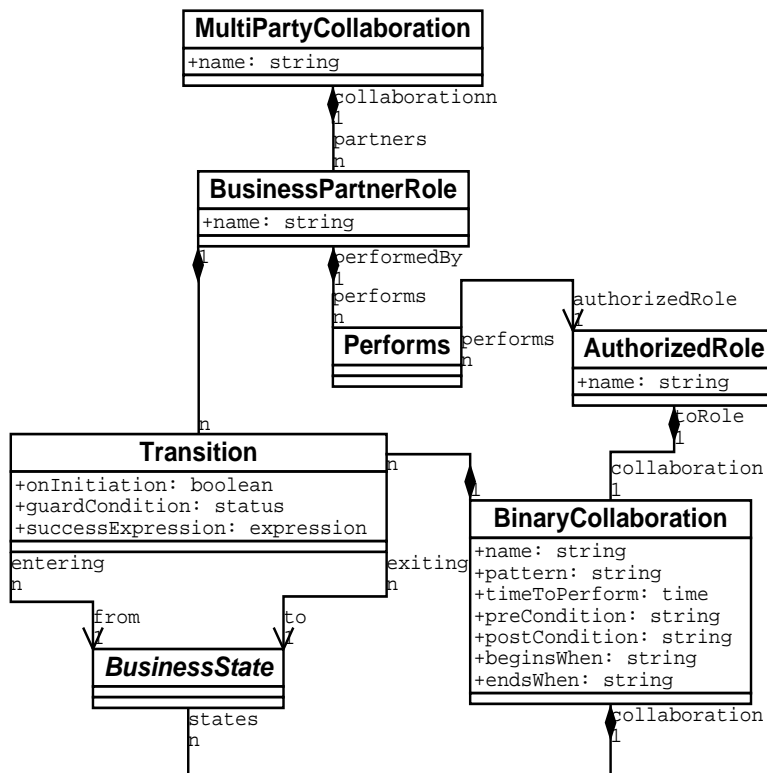


Figure 7: UML Diagram of a Multiparty Collaboration (Adapted from the Business Process Specification Schema document)

In the source code below a Multiparty Collaboration instance is shown. In the example code the reference Business Process is called DropShip. There are three parties involved in this Business Collaboration. The first party/partner is a Customer, the second a Retailer and the third DropShip Vendor. The code reveals how the different roles are taken by which partner. As mentioned above a Transaction always has a requesting role and a responding role.

```

<MultiPartyCollaboration name=''DropShip''>
  <BusinessPartnerRole name=''Customer''>
    <Performs initiatingRole=
      '//binaryCollaboration[@name=''Firm Order'']
      /InitiatingRole[@name=''buyer'']'>
  </BusinessPartnerRole>
  <BusinessPartnerRole name=''Retailer''>
    <Performs respondingRole=
      '//binaryCollaboration[@name=''Firm Order'']
      /RespondingRole[@name=''seller'']'>
    <Performs initiatingRole=
      '//binaryCollaboration[@name=''Product Fulfillment'']
      /InitiatingRole[@name=''buyer'']'>
  </BusinessPartnerRole>
  <BusinessPartnerRole name=''DropShip Vendor''>
    <Performs respondingRole=
      '//binaryCollaboration[@name=''Product Fulfillment'']
      /RespondingRole[@name=''seller'']'>
  </BusinessPartnerRole>
</MultiPartyCollaboration>

```

Next follows the specification of the Choreography.

3.4.5 Specification of the Choreography

Choreography is used to define which Business Transaction follows which Business Transaction. The realisation of the choreography is done with the Transition class. A Binary Collaboration has several Transitions. And a Transition has a “from” Business State and a “to” Business State. The UML diagram for the Choreography in figure 8 shows the abstract Business State class. The classes inheriting from the Business State class are a Start class, a Completion class (Success and failure class), a Fork class, a Join Class and most important a Business Activity class. Obviously the Start state is the first state of a Business Collaboration and the Completion state (success or failure) the last state of a Business Collaboration. Between start and completion, we have the different Business Activities. The Fork and Join classes are used for “if then else” cases. Depending on the outcome of an activity the next state will be different and a different transition occurs. Each Transition has a condition guard and a condition expression.

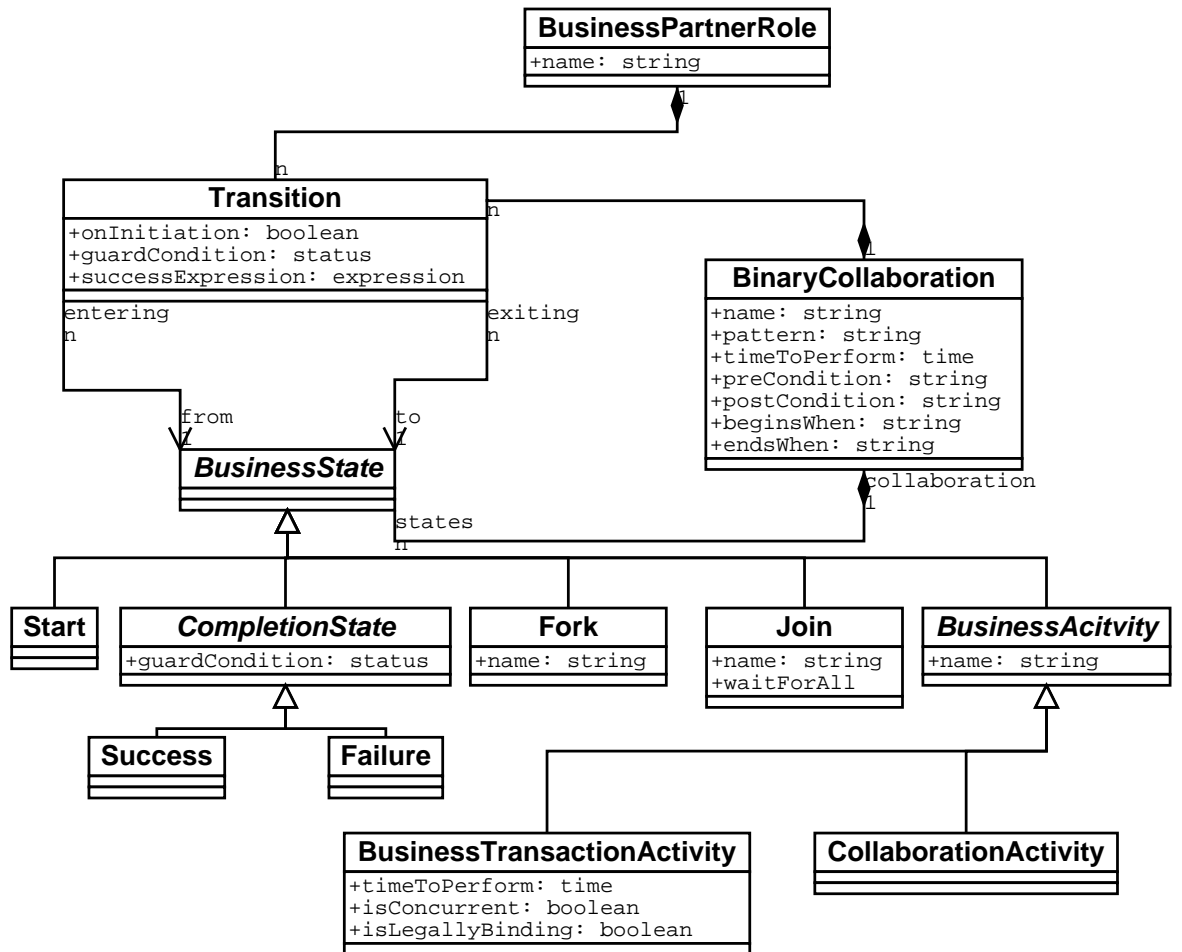


Figure 8: UML Diagram of a Choreography (Adapted from the Business Process Specification Schema document)

The next XML code shows a Binary Collaboration with two Business Transaction Activities. The transition tag defines the transition between the “from” state and the “to” state. The Success and Failure tags the outcome of the Binary Collaboration. Within these tags the condition guards are placed.

```
<BinaryCollaboration name=''Product Fulfillment'' timeToPerform=''P5D''>
  <Documentation>
    timeToPerform = Period: 5 days from start of transaction
  </Documentation>
  <InitiatingRole name=''buyer''/>
  <RespondingRole name=''seller''/>
  <BusinessTransactionActivity name=''Create Order''
    businessTransaction=''Create Order''
    fromAuthorizedRole=''buyer''
    toAuthorizedRole=''seller''/>
  <BusinessTransactionActivity name=''Notify shipment''
    businessTransaction=''Notify of advance shipment''
    fromAuthorizedRole=''buyer''
    toAuthorizedRole=''seller''/>
  <Start toBusinessState=''Create Order''/>
  <Transition
    fromBusinessState=''Create Order''
    toBusinessState=''Notify shipment''/>

  <Success
    fromBusinessState=''Notify shipment''
    conditionGuard=''Success''/>
  <Failure
    fromBusinessState=''Notify shipment''
    conditionGuard=''BusinessFailure''/>
</BinaryCollaboration>
```

The next XML code show a MultiParty Collaboration. The MultiParty Collaboration includes the business partner roles and defines what the performances are for those roles. Within the performances the Binary Collaborations are referenced including the role name within that Binary Collaboration for the role within the MultiParty Collaboration. Further are the transitions defined within the business partner roles with the Binary Collaboration, the “from” state and the “to” state, which are Binary Collaborations.

```
<MultiPartyCollaboration name=''DropShip''>
  <BusinessPartnerRole name=''Customer''>
    <Performs initiatingRole=
      '//binaryCollaboration[@name=''Firm Order'' ]
      /InitiatingRole[@name=''buyer'' ]'/>
  </BusinessPartnerRole>
```

```

<BusinessPartnerRole name='Retailer'>
  <Performs respondingRole=
    '//binaryCollaboration[@name='Firm Order']
    /RespondingRole[@name='seller']'/>
  <Performs initiatingRole=
    '//binaryCollaboration[@name='Product Fulfillment']
    /InitiatingRole[@name='buyer']'/>
  <Transition
    fromBinaryCollaboration='Firm Order'
    fromBusinessState=
      '//binaryCollaboration[@name='Firm Order']
      /[@name='Create Order']
    toBusinessState=
      '//binaryCollaboration[@name='Product Fulfillment']
      /[@name='Create Order']
  </BusinessPartnerRole>
<BusinessPartnerRole name='DropShip Vendor'>
  <Performs respondingRole=
    '//binaryCollaboration[@name='Product Fulfillment']
    /RespondingRole[@name='seller']'/>
</BusinessPartnerRole>
</MultiPartyCollaboration>

```

3.4.6 The ebXML Business Process Specification Schema UML Class Diagram

All the aspects of Business Collaborations, Business Transactions, Business Document flows and Choreography are realised in a UML class diagram. See figure 9 for this one class diagram. The Business Process Schema Specification describe the mapping between the class diagram and the XML schema code.

There are further details which are not treated here, like the facility to legally bind an ebXML Document, the usage of cryptography for encryption, digital signature and non-repudiation. Further details are error handling in the message flow, like time outs, handling and occurrence of exceptions and reliability. The ebXML Messaging Service chapter 3.7 defines the message transportation typically with header and content based on SOAP with attachments.

3.5 ebXML Collaboration Protocol Profile and ebXML Collaboration Protocol Agreement

The Collaboration Protocol Profile describes/defines the capabilities of a company. This are the Business Processes and the technical details such as transportation (e.g. HTTP, HTTP over SSL, FTP), messaging (e.g. SOAP) and security constraints (e.g. digital signature, certificates).

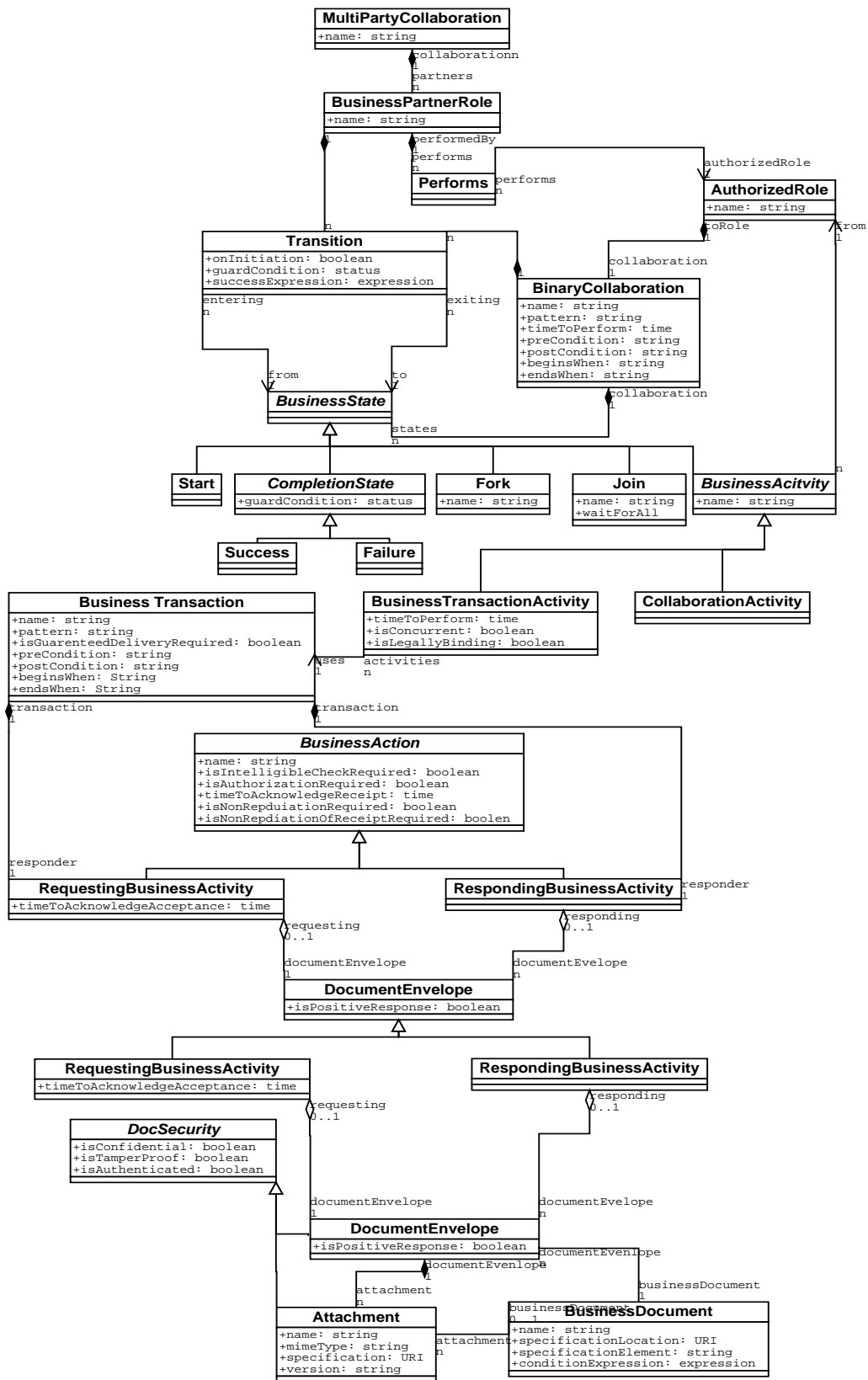


Figure 9: The ebXML Business Process Specification Schema UML Class Diagram (Adapted from the Business Process Specification Schema document)

The Collaboration Protocol Agreement defines the interactions between two parties. The CPA is derived from two or more CPP's. The Collaboration Protocol Profile and Agreement Specification document [9] defines what a CPP and a CPA are composed of, the details of the CPP and CPA. The CPA is like the contract between two parties to conduct electronic business in that defined way.

Again the CPP can reference several Process Specifications (Business Processes or Business Collaborations).

A good question is how to get a CPA from two CPP's. ebXML proposes possible ways and it may not be fully automated yet. A future goal of the ebXML standard is to get a CPA generated automatically. ebXML proposes to calculate the intersection of two CPP's.

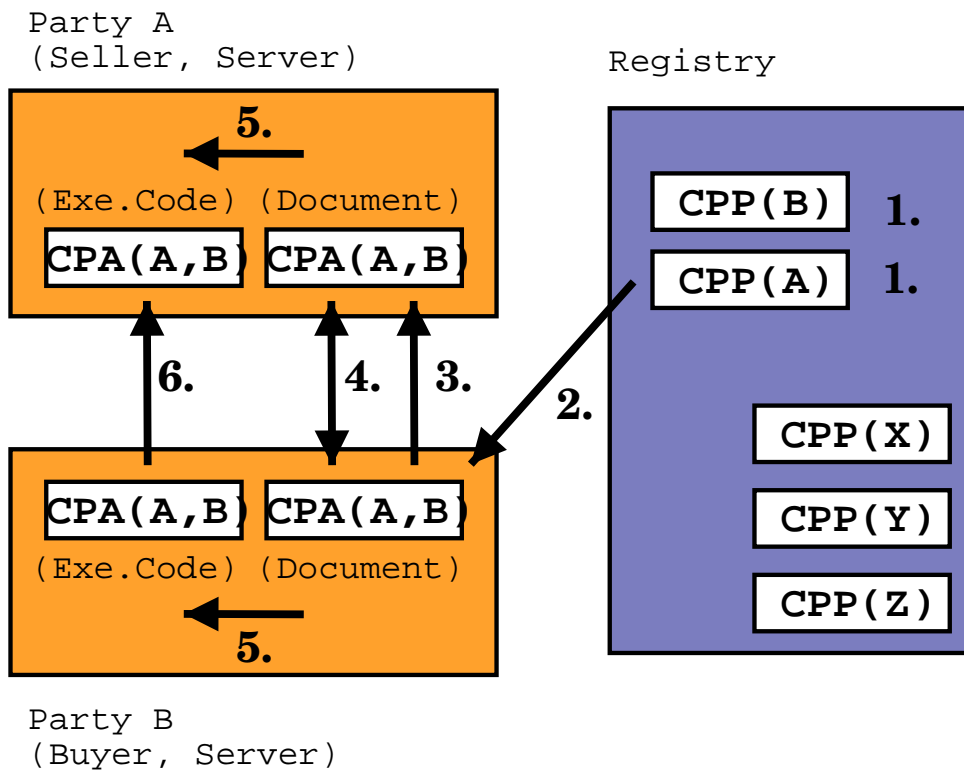


Figure 10: Collaboration-Protocol Profile and Agreement Specification (Adapted from the Collaboration Protocol Profile and Agreement Specification document)

Figure 10 shows a typical scenario. Typically two companies first describe their business capabilities in the form of Business Collaborations. Then they create CPP's by adding information about the company (name, contact etc) and technical information like the desired transport protocol, transport security protocol, and messaging protocol. A CPP *only* references

Business Collaborations stored in the ebXML repository. Then both companies submit their CPP's to the ebXML registry which stores the CPP's in the repository (number 1). A company might then query the ebXML registry to get a suitable CPP from a potential trading partner. The company then calculates the intersection of the retrieved CPP with its own CPP which results in a CPA (number 2). The company sends the calculated CPA to the potential trading partner (number 3) and if the potential trading partner is happy with it he/she accepts the new CPA (number 4). The CPA can be registered at the ebXML registry. In the next step both companies configure their Business Service Interface (BSI) software with the newly created CPA (number 5). After the configuration the BSI software is able to conduct electronic business (number 6). The BSI software *knows* which documents are needed, the choreography of the documents, the transportation protocol, the structure of the individual documents, the role in a business transaction. The CPA defines the conversation between the two trading partners.

What the BSI software does not know is the *content* of the documents. This is very central and not part of the ebXML standard. Software vendors will provide software to connect BSI software to the back-end software systems.

3.5.1 Content of a CPP

The CPP has two layers: the Process Specification layer and the delivery layer. The Process Specification layer references one or more Business Collaborations (Binary or Multiparty Collaborations). The delivery layer is composed of the document exchange layer and the transport layer.

The main CPP structure (defined in XML) has four elements: These are the PartyInfo (information about the organisation), Packaging (), ds:Signature (the digital signature which signs the CPP) and a Comment (general comment) element.

```
<CollaborationProtocolProfile
  ...      <!-- namespace information -->
  >
  <PartyInfo> <!-- one or more -->
  ...
  </PartyInfo>
  <Packaging id=''ID''> <!-- one or more -->
  ...
  </Packaging>
  <ds:Signature> <!-- one or more -->
  ...
  </ds:Signature>
  <Comment>comment</Comment> <!-- one or more -->
</CollaborationProtocolProfile>
```

PartyInfo and Packaging are the more interesting elements and are shown here:

- The PartyInfo element

A sample code fragment illustrates this element. A short overview of the child elements is added after the example code.

```
<PartyInfo>
  <PartyID type=''...''>
    ...
  </PartyID>
  <PartyRef xlink:type=''...'', xlink:href=''...''/>
  <CollaborationRole> <!-- one or more -->
    ...
  </CollaborationRole>
  <Certificate> <!-- one or more -->
    ...
  </Certificate>
  <DeliveryChannel> <!-- one or more -->
    ...
  </DeliveryChannel>
  <Transport> <!-- one or more -->
    ...
  </Transport>
  <DocExchange> <!-- one or more -->
    ...
  </DocExchange>
</PartyInfo>
```

- PartyID

This is a unique id for each company. This can be a D-U-N-S number or a UCC/EAN company code or any other industry related identification mechanism.

- PartyRef

This is a pointer to more information about the company. This can be the company's web site, a UDDI reference or another ebXML registry entity.

- CollaborationRole

This is a very important element and needs further information. Here the content of this element:

```
<CollaborationRole id=''N11''>
  <ProcessSpecification name=''BuySell'' version=''1.0''>
  </ProcessSpecification>
  <Role name=''buyer'' xlink:href=''...''/>
  <CertificateRef certId=''N03''/>
  <ServiceBinding name=''some process'' chanelId=''N02'' packageId=''N06''>
```

```

        <Override action='OrderAck' channelId='N05' packageId='N09'
        xlink:type='simple'
        xlink:href='...' />
    </ServiceBinding>
    <!-- first alternate binding >
    <ServiceBinding chanelId='N04' packageId='N06'>
        <Override action='OrderAck' channelId='N05' packageId='N09'
        xlink:type='simple'
        xlink:href='...' />
    </ServiceBinding>
</CollaborationRole>

```

- Certificate
This identifies the certificates used by this party for security functions.
- DeliveryChannel
This defines the transport and message protocol used for message exchange.
- Transport
This defines details of the supported transport protocols (mentioned in the DeliveryChannel element) used for the message passing.
- DocExchange
This element describes properties of the messaging service.

For further details the specifications are useful or the source code of the open source project open ebXML, at www.openebxml.org

- The Packaging element
The most important parts of the CPP are shown here. There are further XML elements and attributes not listed here.

```

<Packaging id='id'>
    <ProcessingCapabilities parse='...' generate='...' />
    <SimplePart id='id' mimetype='type' />
    <CompositeList>
        <Composite mimetype='type' id='name'
            mimeparameters='parameter'>
            <Constituent idref='name' />
        </Composite>
        <Encapsulation mimetype='type' id='name'>
            <Constituent idref='name' />
        </Encapsulation>
    </CompositeList>
</Packaging>

```

The ebXML specifications explain each element in more detail.

3.5.2 Content of a CPA

The CPA is the agreed document between two parties. The content of first level in the XML file is listed below.

```
<CollaborationProtocolAgreement
  ... <!-- namespace information -->
  >
  <Status value='proposed' />
  <Start>1988-04-07T18:39:09</Start>
  <End>1990-04-07T18:40:00</End>
  <ConversationConstraints invocationLimit='100',
    concurrentConversations='4' />
  <PartyInfo>
  ...
  </PartyInfo>
  <PartyInfo>
  ...
  </PartyInfo>
  <Packaging id='N30'> <!-- one or more -->
  ...
  </Packaging>
  <ds:Signature>signature
  </ds:Signature>
  <Comment xml:lang='en-gv'>comment</Comment>
</CollaborationProtocolAgreement>
```

Some information on the individual tags:

- Status
The status of the CPA. Possible values are: proposed, agreed or signed.
- Start and End
The time when this CPA is valid. The lifetime of the CPA.
- ConversationConstraints
This optional element can set constraints.
- PartyInfo
The same element as in the CPP. References a party.
- Packaging
The packaging elements defines how the message is packaged as well as security parts.
- ds:Signature
The digital signature.
- Comment
A simple optional comment.

According to this CPA the trading partners will do business. The ebXML-dev mailing lists⁹ show two benefits of a CPA (discussed in November 2001). First the dynamic finding of trading partners (which might become a more common business scenario in the near future) and the dynamic configuration of the Business Service Interface (BSI) according to the CPA. How the back-end application interacts with the Business Service Interface is not defined in the ebXML specification. This advantage is for companies with more than a few trading partners. The BSI can handle any number of different CPA's and according to the current CPA the BSI acts upon it.

3.6 ebXML Core Components

ebXML puts a lot of effort into re-usability. In the CPP document all kinds of objects get referenced. A CPP references Business Processes. A Business Process references Business Documents. Business Documents reference further entities. The last entity which can get referenced is a Core Component but Core Components can reference Core Components. Such Core Components are called Aggregated Component. Core Components get stored in the ebXML registry. Of course new Core Components can be added and common Core Components can be updated, extended. For this reason Core Components are versioned. Core Components are an English language construct and Core Components are used across different industries.

Core Components have to be implemented first. Actually a core component has to be discovered/defined first (the ebXML Core Component Discovery and Analysis [4] technical report explains more on this). Then it can be defined in XML.

The UBL (Universal Business Language) team has the task to define some Core Components. The UBL group took the CommerceOne xCBL¹⁰ as a basis to start. This is a bottom-up approach and this approach has to meet the top-down approach somewhere in the middle.

3.7 ebXML Messaging Service

Generally speaking the ebXML Messaging Service (ebXML MS) handles the delivery and receiving of the XML messages from point A to point B and is defined in ebXML Messaging Service Specification [11]. ebXML MS is based on the common protocol TCP/IP and on a higher level on protocols like FTP, HTTP, SMTP plus others. Further ebXML MS is based on SOAP Messaging with Attachments (Simple Object Access Protocol)¹¹

ebXML calls the software which deals with the messaging the Message Service Handler (ebXML MSH). The ebXML MSH is an entity which is used

⁹<http://www.ebxml.org>

¹⁰<http://www.xcbl.org>

¹¹currently at <http://www.w3.org/TR/SOAP/>

by the Business Service Interface to finally send and receive messages.

The ebXML Messaging Service Team divided their work into the packaging specification, ebXML SOAP extensions, message service handler services, reliable messaging, error reporting and handling and security.

3.7.1 Packaging Specification

The Packaging Specification deals with how the data has to be organised/packaged. Figure 11 shows the ebXML Message Structure. An ebXML Message Package has basically a Header Container and zero or more Payload Container. The Header Container has ebXML and SOAP specific information whereas the Payload Container has the “real” data of the message.

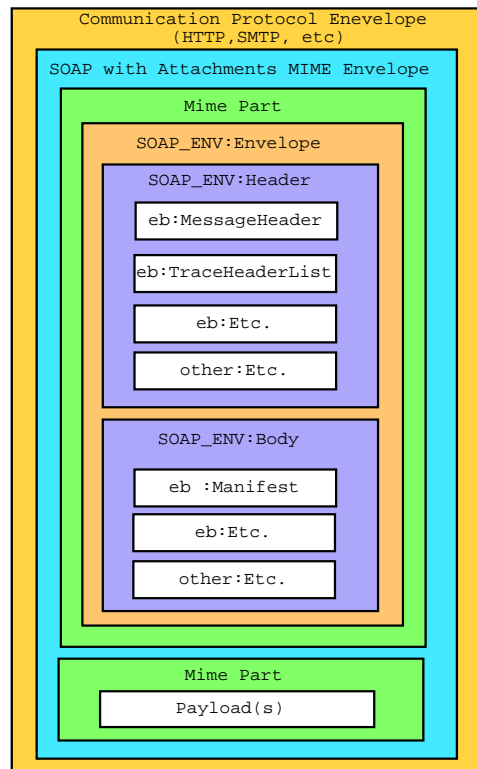


Figure 11: The ebXML Message Structure (Adapted from the ebXML Messaging Service Specification)

The Header Container has two parts, a SOAP Header and a SOAP Body. A simple SOAP Envelope is listed below.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV='''...''>
  <SOAP-ENV:Header> ... </SOAP-ENV:Header>
  <SOAP-ENV:Body> ... </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.7.2 ebXML SOAP Extensions

ebXML now adds elements to the SOAP Header and SOAP Body elements. The SOAP Header gets the following elements: MessageHeader, TraceHeaderList, ErrorList, Signature, Acknowledgment and Via. The SOAP Body gets these elements: Manifest, StatusRequest, StatusResponse, DeliveryReceipt. These XML elements are listed here.

In the SOAP Header are:

- MessageHeader

The MessageHeader element goes into the SOAP Header element and has the following child elements: From, To, CPAId, ConversationId, Service, Action, MessageData, QualityOfServiceInfo, SequenceNumber, Description. The MessageHeader element has a further two attributes: mustUnderstand and Version. This element plays an important role. The child elements of the MessageHeader will be described next.

The elements often have further child elements. It is worth noting that some of the elements are introduced to realise the functionality of security, error handling, reliable messaging and might be confusing at the moment. It is also worth pointing out, that some elements are related to other elements, e.g. if one element is used another element must be used. A * indicates if the element is mandatory or not.

- From and To *

These two elements seem clear. They define the sender of the message and the receiver of the message. This element has the PartyId which can contain an internationally known identifier, like a DUNS number or a simple web site address.

- CPAId *

This element is also quite obvious. The element references the CPA of the two trading partners.

- ConversationId *

This element represents a unique ID for the current conversation between two trading partners. The initiating MSH creates this ID and from that moment this ID will be used in the proceeding messages.

- Service *

The Service element relates back to the Business Process Specification, where authorised roles within a business process are used. The service element names the activity of a party for this message. For example a service could be: “SupplierOrderProcessing”.

- Action *
This element identifies a process within the Service. This could be a “NewOrder” or Acknowledgment of reliable messaging purpose.
- Messagedata
This element has further child elements: MessageId, Timestamp, RefToMessageId, TimeToLive.
 - * MessageId
It is obvious that each Message has its own identification for further referencing.
 - * Timestamp
It is also obviously the Timestamp when the message was created.
 - * RefToMessageId
If the message is an Acknowledge Message (used for reliable messaging) this element references a previous message by its ID.
 - * TimeToLive
This element is used for the functionality of reliable messaging and sets a time frame for the delivery of the message.
- QualityOfServiceInfo
This element deals with reliable messaging. The QualityOfServiceInfo element has three child elements: deliverySemantics, messageOrderSemantics and deliveryReceiptRequested. The deliverySemantics has an example value of OnceAndOnlyOnce which indicates the importance of this element. Chapter 3.7.4 has more details of how reliable messaging is achieved within ebXML.
- SequenceNumber (* when used with QualityOfServiceInfo)
This element ensures that the order of messages is recognised.
- Description
This is a simple human readable description of the message and is optional.

Here is a sample Header.

```
<eb:MessageHeader id=''...' eb:version=''1.0'' SOAP-ENV:mustUnderstand=''1''>
  <eb:From>
    <eb:PartyId>uri:example.com</eb:PartyId>
  </eb:From>
  <eb:To eb:type=''someType''>QRS543</eb:To>
  <eb:CPAid>http://www.ebxml.org/cpa/112345</eb:CPAid>
  <eb:ConversationId>934523405</eb:ConversationId>
  <eb:Service eb:type=''myservicetype''>QuoteToCollect</eb:Service>
  <eb:Action>NewPurchaseOrder</eb:Action>
```

```

    <eb:MessageData>
      <eb:MessageId>mid:UUID-2</eb:MessageId>
      <eb:Timestamp>2001-12-06ST12:39:05</eb:Timestamp>
      <eb:RefToMessageId>mid:UUID-1</eb:RefToMessageId>
    </eb:MessageData>
    <eb:QualityOfServiceInfo
      eb:deliverySemantics='OnceAndOnlyOnce'
      eb:deliveryReceiptRequested='Signed' />
  </eb:MessageHeader>

```

More elements in the SOAP Header:

- **TraceHeaderList**
The TraceHeaderList element consists of one or more TraceHeaders elements. If a message is transferred to a remote ebXML MSH via multiple MSH, each MSH adds a TraceHeader to the TraceHeaderList. It is typically used in a multi-hop scenario.
- **ErrorList**
The ErrorList element has one or more Error elements and indicates that the message referenced by RefToMessageId has an error. The ErrorList is used to indicate errors. Check 3.7.5 for more details.
- **Signature**
There can be zero or more Signature elements which are digital signatures. See 3.7.6 for more on digital signatures.
- **Acknowledgment**
The Acknowledgment element is set when the purpose of the message is to acknowledge of a received message. The ID of the received message is put into the RefToMessageId of the Acknowledgment element.
- **Via**
The Via element indicates the way a message goes from point A to point B. The Via element has several child elements or attributes: syncReply, reliableMessagingMethod, ackRequested, CPAId, Service and Action.

In the SOAP Body element are:

- **Manifest**
The Manifest element is a very important element. The Manifest element has one or more Reference elements. As mentioned above an ebXML Message has a Header Container and zero or more Payload Container. Each Reference element references one payload in the Payload Container. The Reference element itself has a Schema element which can link to the schema of the referenced payload. The referencing is done via xlink (see 2.8.2 for some information on xlink).

- **StatusRequest**
The StatusRequest element is used to request status of a message, referenced by the RefToMessageId element of the StatusRequest element.
- **StatusResponse**
The StatusResponse element is the return message to a StatusRequest message. This element has child elements and attributes: RefToMessageId, TimeStamp, version, messageStatus and an id. The MessageStatus can be: Unauthorized, NotRecognized or Received.
- **DeliveryReceipt**
The DeliveryReceipt element is used to indicate, that a previous message (referenced by RefToMessageId) was received.

3.7.3 MSH Services

There are two supporting MSH Services: Message Status Request and Message Service Handler Ping. The Message Status Request Service simply handles the request-response of messages. The Message Service Handler Ping service is simply to test if MSH is accessible. For security reasons a responding Pong can be omitted.

3.7.4 Reliable Messaging

Reliable Messaging is responsible, for a message primarily getting from A to B and that that message only gets once to B, plus that the messages are in order. To achieve this functionality there are several XML elements in the CPA plus in the SOAP Message Header. If the element reliableMessaging-Method is set to ebXML, then the MSH has to take control of the reliability. This is achieved by the ackRequested element. Basically each message has to get acknowledged.

3.7.5 Error Reporting and Handling

There are three types of error which can happen: ebXML/SOAP messages are deordered, reliable messaging failures and security problems. The specification makes sure, that an error is propagated to the right place.

3.7.6 Security

Security has an important role in the design of ebXML. The common security risks are: Unauthorised access, data integrity and confidentiality and denial-of-service or spoofing.

The W3C XML Signature specification is used for digital signatures and XML Encryption for encryption. Certificates are used to do signatures.

It is worth noting, that it should be possible to edit the SOAP Header, e.g. the TraceHeaderList, while the rest of the message is encrypted.

4 Definition of Business Processes

Chapter 3.4 dealt with Business Process related topics in particular with the runtime aspects of the ebXML framework. This chapter deals more with the business part behind Business Processes. UMM is a modelling methodology which allows modelling the details of a Business Processes.

The Business Process and Business Information Analysis Overview technical report [6] shows again how to get from basic Business Process Modelling to a running system. It emphasises once again the “Collaboration Business Processes”, not just a company doing its internal administration electronically but seeing the big picture from an economical view point (inter-enterprise). The report sees “buying and selling products or services” as the most common business collaboration and “order-fulfilment-payment” as the most common pattern for this most common business collaboration .

People thought about Business Processes long before ebXML and as a result there are already models available which have to be converted to be ebXML conform. This provides a good starting point to re-use previous works from different standardisation organisations. The Business Process and Business Information Analysis Overview technical report [6] mentions the Resource-Event-Agent (REA) Enterprise Ontology as an archive of pre-defined business processes and business process patterns. Unfortunately at the date of the writing of this report the official web site of REA ¹² was not operational.

Once again, the idea is to populate the Registry/Repository with meta models, Core Components, Business Information Objects, Business Documents etc. Once the Registry/Repository is populated, new companies, organisations can browse the Registry/Repository and should be able to just “click” their CPP together. ebXML talks about Business Process Editors and supporting software to enable this approach. It really depends what a user of ebXML is after. Whether the user wants to add new Core Components, analyse new Business Processes or simply re-use stored data/models. The future will show how ebXML will be accepted and how much the Registry/Repository gets populated. ebXML sees the possibility to have not only public Meta Models (Core Components, Business Information Objects, etc) but also private ones. If the definition of a new killer Business Process wants to be kept private and only selected companies/organisations are allowed to re-use the new killer Business Process, this can be achieved with private, not public available Meta Models. Of course for Small to Medium Enterprises (SME) the more public available information is provided the bigger the chances they are part of the global electronic marketplace, which in the long run can be important to big companies with many SME suppliers and many SME customers.

¹²<http://www.reamodel.org>

4.1 Analysis Worksheets for modelling Business Processes

For a new industry to get their Business Process defined, there are Analysis Worksheets available to follow. Teams who have to model Business Processes shall use the Business Process Analysis Worksheets & Guidelines technical report [5]. Once again, it is not certain, that a company finds its specific Business Processes available in the Registry. In that case the company (or better the industry which this company is part of) has to define the company/industry specific Business Processes. Different ebXML Documents talk about the process to analyse businesses in order to get artifacts for ebXML. The approaches are very often closely related to the UMM modelling methodology.

4.2 Common Business Processes

The Core Components and Business Process Analysis Team worked in the evaluation of common Business Processes. To categorise the different Business Processes the Porter Value Chain is taken as a categorisation schema. Figure 12 shows the value chain after Michael E. Porter.

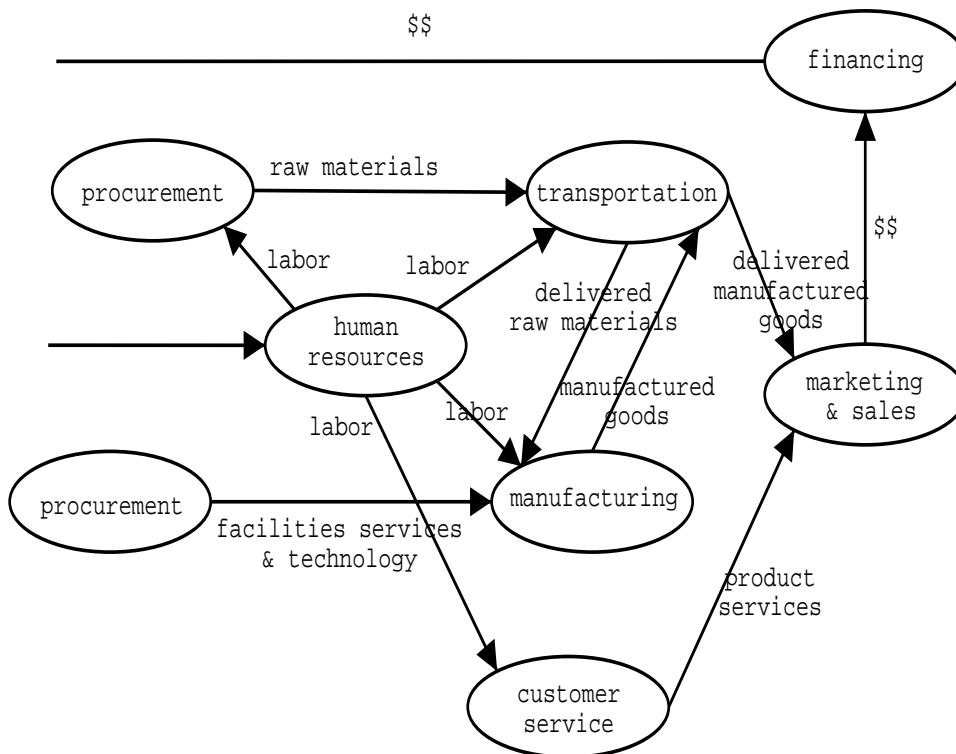


Figure 12: Graphical representation of the Porter Value Chain (Adapted from the ebXML Catalogue of Common Business Processes)

Even if this Porter Value Chain is dated back to 1980, common categories like human resources, customer service, procurement, manufacturing, financing, sales can be found and parallels to electronic business in general as introduced in chapter 2.1 are obvious.

The ebXML Catalogue of Common Business Processes v1.0 [8] lists around 200 common business processes in the following categorisations: Procurement Management, Administration, Product Introduction, Financial, Transportation and Logistics, Marketing Management, Product Development, Product Planning and Service Support. As input for the list several standards were taken into account: EDIFACT, X.12, RosettaNet Partner Interface Protocol, CII, OAG BOD's, xCBL and probably more.

5 Conclusion

The public ebXML specifications provide the necessary information to understand how ebXML provides a framework for a global electronic marketplace.

5.1 Summary

ebXML is a very promising effort to create a global electronic marketplace. It has to be seen as a future and long term goal to get many companies on the electronic business bandwagon. The ebXML specification needs time to be adopted by companies. Many companies are doing electronic inter-enterprise business today not only with EDI but with the availability of the Internet, and it is a matter of time before these companies embrace ebXML. ebXML is used to electronically exchange business documents which focus on collaboration.

The ebXML Registry/Repository Specification, ebXML Business Process Specification Schema, ebXML Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA) specifications, ebXML Core Components (CC) and ebXML Messaging Service (MS) Specification are the main parts of ebXML specifications.

The ebXML Registry/Repository is the distributed database of ebXML. The ebXML Registry/Repository provides an interface for clients to register ebXML Collaboration Protocol Profiles, Collaboration Protocol Agreements, Core Components, Business Objects and many more. It also provides an interface to query objects.

The ebXML Business Process Specification Schema defines how Collaboration is modelled with Business Processes. An ebXML Collaboration is not limited to two trading partners. A Multiparty Collaboration involves more than two trading partners (e.g. supplier, shipping company, customer). A Multiparty Collaboration is broken down to Binary Collaborations between two trading partners. In a Binary Collaboration each trading partner has a role (e.g. buyer, seller). A Collaboration is composed of several Transitions. Each Transition has an entering Business State and an exiting Business State. A Business State can be either Start, Completion (either success or failure), Fork, Join or a Business Activity. The Business Activity is a specific Transaction. Each Transaction has a Request and a Response. Each Request has one or more Business Documents attached. The schema of a specific Business Process will be stored in the ebXML Registry/Repository. Each company participating in the ebXML world will register one or more ebXML CPP's.

An ebXML CPP references one or more Business Processes stored in the ebXML Registry/Repository. A CPP also has ebXML technical information (type of security, transportation protocol etc) associated with it. The CPA

is a derived document from two CPP's. Two companies agree upon a CPA and configure their ebXML compliant software system (Business System Interface) with the CPA.

ebXML Core Components are entities which occur among different industries and are low level data fields (e.g. address, location, zip). Business Objects are aggregations of Core Components. Business Documents (e.g. order, invoice) are composed of Business Objects and single Core Components. All these documents also get stored in the ebXML Registry/Repository.

The ebXML Messaging Service Specification defines the messaging service of ebXML documents. ebXML Messaging Service enables security, error handling and reliable messaging.

5.2 ebXML Problems

A major goal of ebXML was to enable Small to Medium Enterprises to become part of the global electronic marketplace. As Rawlins points out at <http://www.rawlinseconsulting.com> a major problem lies in the integration of the ebXML Business Service Interface within the enterprise business applications. It is important to notice, that ebXML is business process based where a business process has several states. The business applications of Small to Medium Enterprises will have difficulty to integrate the ebXML Business Service Interface. Further some business processes are truly industry specific and those will have problems to integrate them with their low end ERP systems because the low end ERP system vendors are less likely to implement ebXML. Further concrete problems are:

- There is no formal description on how to derive a CPA from two CPP's. The ebXML specification states that this process may not be 100% automated and human interaction may be necessary.
- There are no concrete conversion rules for UML to XML provided by ebXML.
- There is a problem of how to find a Registry/Repository. It seems that there will be many different Registries/Repositories for different industries. The usage of UDDI (Universal Description, Discovery and Integration ¹³) can be a solution to this problem.
- Companies cannot start using ebXML yet because the Registry/Repository is still empty. It needs time to get the Core Components stored in the Core Library (Registry/Repository), then to create the Business Information Objects which are needed for the Business Documents. It needs time to create Common Business Processes and store them in the Business Library (Registry/Repository). The UBL (Universal

¹³<http://www.uddi.org>

Business Language) team is at the work to create the Core Components defined in XML.

- Today's business applications are not ebXML ready yet. This is especially true for software applications of the Small to Medium Enterprise (SME) market. It needs time until these software packages integrate ebXML and allow the SME's to do ebXML business transparently. Software companies committed their will to start the integration of ebXML within their products.

The time will come and the problems will be addressed or are already underway to be solved. ebXML needs time to evolve and to provide a solid solution for a global electronic marketplace to allow inter-enterprise businesses.

5.3 Possible future directions

In order to establish the ebXML standard the problems mentioned in chapter 5.2 have to be solved. For new companies it is important, that the ebXML Registry/Repository has content. New companies want to use Business Process Editor tools to create a suitable Collaboration Protocol Profile (CPP). The specific industries have to provide the common Business Processes, Core Components to enable ebXML for its members. Industry wide organisations have to become aware of ebXML and start to provide the ebXML Registries with data.

Seamless automation for new companies has to be provided. The creation of a CPP has to be made as easy as possible. ebXML has to clearly define how to create a CPA from two separate CPP's.

The free software / open source community can create cost effective, reliable, secure, open, free and standard compliant ebXML software components.

A APPENDIX List of the official ebXML Documents

Table 1 shows all available major documents including their current filename from the <http://www.ebxml.org> web site. Current state is May 2001.

B APPENDIX Relating free software / open source Projects

Here are descriptions of two projects of the free software / open source community which can help to bring ebXML to Small to Medium Enterprises (SME's).

B.1 openebXML - open ebXML Implementation

The openebXML project at <http://www.openebxml.org> is a open source project to implement the ebXML specifications under a Mozilla like licence.

The openebXML project also has a couple of different tools available today but needs more development to evolve. openebXML tries to cover the full range of the different main parts of the ebXML specification. There are other open source projects which implement parts of ebXML parts (e.g. there is an project for the ebXML Registry, as well is there a Java API specification for the ebXML Registry).

The current projects at openebXML.org include BML, a binary markup language, a Workbench for edition business definitions, a Business Process Server called Red-Line an ebXML Registry, an ebXML Message Handler and some further tools. Some of the subprojects are still in the inception phase or elaboration phase and need much more development.

The latest tool at openebXML is a CPP (Collaboration Protocol Profile) verification tool, which checks a CPP for validity and references of Business Processes at an ebXML Registry. This tool is not available at the web site yet (December 2001).

B.2 GNUe - GNU Enterprise

GNUe is a free software project with the idea to implement a Enterprise Resource Planning system. With such a free system Small to Medium Enterprises can afford to go electronically without a specific vendor lock-in.

Below is the new draft of the official GNUe web site at <http://www.gnuenterprise.org>

OVERVIEW

GNU Enterprise (GNUe) is a suite of tools and applications for solving the needs of the enterprise. From human resources, accounting, customer

File Name	Document Name
	Technical Specifications:
ebBPSS.pdf	Business Process Specification Schema v.1.0.1
ebTA.pdf	ebXML Technical Architecture Specification v.1.04
ebRIM.pdf	Registry Information Model v.1.0
ebRS.pdf	Registry Services Specification v.1.0
ebREQ.pdf	ebXML Requirements Specification v.1.06
ebCCP.pdf	Collaboration-Protocol Profile and Agreement Spec v.1.0
ebMS.pdf	Message Service Spec v.1.0
	Technical Reports:
bpOVER.pdf	Business Process and Business Info Analysis Overview v1.0
bpWS.pdf	Business Process Analysis Worksheets & Guidelines v1.0
bpPATT.pdf	E-Commerce Patterns v1.0
bpPROC.pdf	Catalogue of Common Business Processes v1.0
ccOVER.pdf	Core Component Overview v1.05
ebCCD&A.pdf	Core Component Discovery and Analysis v1.04
ebCNTXT.pdf	Context and Re-Usability of Core Components v1.04
ccCTLG.pdf	Guide to the Core Components Dictionary v1.04
ebCCNAM.pdf	Naming Convention for Core Components v1.04
ebCCDOC.pdf	Document Assembly and Context Rules v1.04
ccDRIV.pdf	Catalogue of Context Drivers v1.04
ccDICT.pdf	Core Component Dictionary v1.04
ccSTRUCT.pdf	Core Component Structure v1.04
secRISK.pdf	Technical Architecture Risk Assessment v1.0
	Reference Material:
ebGLOSS.pdf	ebXML Glossary
	White Papers:
bpTAREV.pdf	Proposed revisions to Technical Architecture Spec. v1.04
rrUDDI.pdf	Using UDDI to find ebXML Registry/Repository
secREG.pdf	ebXML Registry Security Proposal

Table 1: Current ebXML available documents from <http://www.ebxml.org>

relationship management and project management to supply chain or e-commerce, GNUe can handle the needs of any business, large or small. If you are looking for a full-function ERP, GNUe is the package for you.

Beyond applications, GNUe is a development framework that enables enterprise information technology professionals to customize applications for their businesses. The GNUe platform boasts an Open Architecture and easy maintenance. It gives users a modular system and freedom from being stuck with a single-source vendor. Plus, users get consistency and the ability to tap into a network of best practices from other enterprises, saving valuable development time.

GNUe is a Free Software project with a corps of volunteer developers around the world working on GNUe projects. This provides the added benefits of easy internationalization of applications. The project is working to provide a worldwide GNUe community, allowing everyone who is involved in the project access to other talented business information technology professionals.

GNUe supports multi-currency processing (including euro support) and multi-language interfaces.

GNUe is a work in progress.

FREE SOFTWARE

GNU Enterprise is released under the GPL license. For an example, please see the GPL license at <http://www.gnu.org/copyleft/gpl.html>. The actual license covering GNU Enterprise is included in all GNU Enterprise distributions. For a description of why GPL and why free software see <http://www.gnu.org/philosophy/why-free.html>.

OPEN ARCHITECTURE

Open Architecture means you can choose which applications you wish to use; you don't have to install the entire package. Or, you can develop your own applications and integrate them. GNUe is built around open standards. It is also designed from the ground up as a modular system and utilizes the latest in object technology. Object technology means that parts of the overall system may be improved and capabilities added without monolithic or system wide changes. For example, you may chose to implement GNUe's Accounts Receivable but keep your existing Order Entry/Invoicing.

ERP

Enterprise Resource Planning (ERP)

A comprehensive, integrated set of business solutions which helps your company gain a competitive edge by integrating all business processes and optimizing the usage of the resources you have available.

The project is still in its beginning and needs development. The main application server is called GEAS and has already the possibility to use

Business Objects. The other tool of GNUe are the GNUe-Forms which provide a user interface to an abstraction (GNUe-Common tool) of some supported databases.

B.3 GNU Free Documentation License

GNU Free Documentation License
Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with

modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to

the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original

- publisher of the version it refers to gives permission.
- K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
 - L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
 - M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
 - N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

References

- [1] Daniel Amor. *THE E-business (R)EVOLUTION*. Prentice Hall PTR, 2000.
- [2] Hossein Bidgoli. *Electronic Commerce Principles and Practice*. Academic Press, 2002.
- [3] Vivek Choudhury, Kathleen S. Hartzel, and Benn R. Konsynski. Uses and consequences of electronic markets: an empirical investigation in the aircraft parts industry. *MIS Quarterly*, 22(i4):471, december 1998.
- [4] ebXML.org. Core component discovery and analysis v.1.0.4. <http://www.ebxml.org/specs/ebCCDA.pdf>, May 2001.
- [5] ebXML.org. ebxml business process analysis worksheets & guidelines v.1.0. <http://www.ebxml.org/specs/bpWS.pdf>, May 2001.
- [6] ebXML.org. ebxml business process and business information analysis overview v.1.0. <http://www.ebxml.org/specs/bpOVER.pdf>, May 2001.
- [7] ebXML.org. ebxml business process specification schema v.1.0.1. <http://www.ebxml.org/specs/ebBPSS.pdf>, May 2001.
- [8] ebXML.org. ebxml catalog of common business processes v.1.0. <http://www.ebxml.org/specs/bpPROC.pdf>, May 2001.
- [9] ebXML.org. ebxml collaboration-protocol profile and agreement specification v.1.0. <http://www.ebxml.org/specs/ebMS.pdf>, May 2001.
- [10] ebXML.org. ebxml e-commerce patterns v.1.0. <http://www.ebxml.org/specs/bpPATT.pdf>, May 2001.
- [11] ebXML.org. ebxml message service specification v.1.0. <http://www.ebxml.org/specs/ebMS.pdf>, May 2001.
- [12] ebXML.org. ebxml registry information model v.1.0. <http://www.ebxml.org/specs/ebRIM.pdf>, May 2001.
- [13] ebXML.org. ebxml registry services specification v.1.0. <http://www.ebxml.org/specs/ebRS.pdf>, May 2001.
- [14] ebXML.org. ebxml technical architecture specification v.1.0.4. <http://www.ebxml.org/specs/ebTA.pdf>, May 2001.
- [15] Ravi Kalakota and Marcia Robinson. *E-Business:roadmap for success*. Addison Wesley, first edition, 1999.

- [16] Alan Kotok and David R.R Webber. *ebXML The New Global Standard For Doing Business Over The Internet*. New Riders Publishing, first edition, September 2001.
- [17] Mark Norris, Steve West, and Kevin Gaughan. *eBusiness Essentials*. John Wiley & Sons, Ltd, 2000.