MATCHING OF EBXML BUSINESS PROCESSES

Document Details:	
Author	Dennis Krukkert
Document Number	Report
Project number	IST 2001-28548 openXchange
Deliverable type	Research report
Version	0.31
Contractual Delivery date	July 2003
Actual Delivery date	-
Status	In Progress

Summary: This document describes the results of a M.Sc thesis research on matching of ebXML business processes.

Distribution

Consortium

File

TABLE OF CONTENTS

EXECUTIVE SUMMARY

EbXML is a relative new standard that presents a framework for doing electronic business. One of the things that distinguishes ebXML from other standards is the possibility for a company to specify its business processes and publish this in a public registry. When a company is looking for a business partner, it can compare the business process specification of itself with those of others. When business processes become more and more complex and the number of process specifications increases, it becomes more and more time consuming to manually compare the process specifications.

This report presents the results of an effort to create a system for automated business process matching within ebXML. The matching process has two aspects: the matching of structure and the matching of content. The first results are positive and a solution is presented that can match both aspects. Nevertheless, some issues remain unsolved and additional research is required.

LIST OF FIGURES

Komt nog

1 INTRODUCTION

1.1 Background

Every time two companies want to conduct business, communication is required. This communication usually consists of a flow of documents like orders, order conformations, invoices, etc. These documents vary in nature, but they all have one thing in common: they need to be transported from one party to another. Technology has advanced the means of communication, but still, devices like fax, telex and e-mail require human involvement in the communication process. Although human involvement has a lot of advantages, humans have two major drawbacks: they make mistakes and the time needed for a single administrative action is significant larger then the time needed by a computer.

In conventional trade (figure 1.1a), first an employee of the ordering company has to print list off all items he wants to order. Then, through fax, phone or some other form of communication, the supplier must be made aware of the fact that someone is ordering supplies from his. Once an employee of the supplying company communicated with the potential customer, he has to enter the order in their ERP (Enterprise Resource Planning) system. Now, the order can be handled and goods can be delivered.

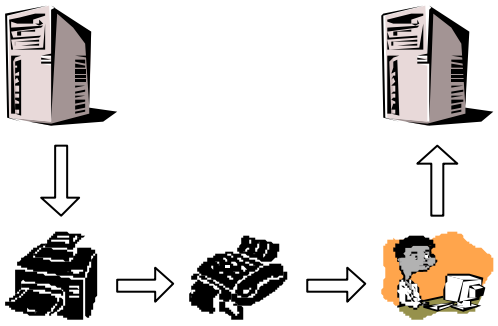


Figure 1.1a: Traditional trade

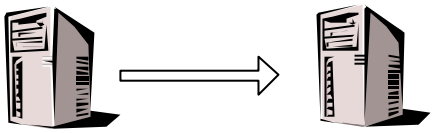


Figure 1.1b: Electronic trade

For cost efficiency and error reduction, it would be better to eliminate human involvement in this part of the process (figure 1.1b). In the '80s the organization currently known as UN/CEFACT worked on a standard for electronic data interchange (EDI) between applications in different companies. The result of this work led to the publishing of the UN/EDIFACT standard by the International Standards Organization (ISO) in 1987: ISO9735. The great advantage of this system seems evident, since the time that documents take to reach their destination (inside the other party's application) are reduced to almost zero. Also, the human factor is eliminated, resulting in fewer errors.

EDI has had (and still has) a lot of impact on the way companies do business, especially large and medium-sized companies and is still being used all over the world. To give an impression: nearly all companies in the Fortune 500 have EDI connections with some of their suppliers [Ram99]. Although EDI has been very successful, there are some major drawbacks to this system. First of all, an EDI system is expensive, not affordable by small or medium sized companies. Money is saved during the transactions, so EDI is only interesting if a company does a lot of transactions. For an EDI system, specialised middleware is needed and a company that implements EDI has to agree upon a message format with each of its trading partners that it wants to use EDI with [Blo92]. The system offers no flexibility towards the message: the format is fixed. Finally, EDI only offers the possibility of sending messages and receiving. Other information, like business profiles or business processes or binary data like pictures cannot be exchanged [Ram99].

In 1999, a join effort of UN/CEFACT and OASIS resulted in the ebXML project [Ebx02]. The main goal of this project was to define a framework for electronic business to business (b2b) e-commerce. The first meeting was in November 1999 and 18 months later, in May 2001, the first version of the framework was completed. The framework consisted of about 25 documents specifying a large variety of concepts including business profiles, business process specification, registry / repository specification, core components and of course document specification. EbXML document specifications are the counterpart of the EDI message standards.

The ebXML project aims to lower the initial costs so that SME's can also participate in the process of electronic business. EbXML has some advantages over EDI. First of all, information is stored (and messages are exchanged) in the XML format, which is extensible and is or will be supported by most standard applications and middleware. Secondly, the Internet is used to exchange documents. Since nowaday most companies are connected to the Internet, a large infrastructure is available.

Companies can make a profile and publish that in a public registry. Their profile can reference business processes, either newly designed or already existing, so others can see how they want to commit business. Communication consists of exchanging xml documents. The way this exchange takes place is very flexible. Companies can use protocols like http, ftp, smtp, etc for the exchange (even floppies could be used technically, but then the argument of time reduction is not valid anymore). The supported protocols can be published in the company's profile.

This research is done as part of the openXchange project that aims to create a framework for doing cross-industry, cross-country e-business. At the start of the openXchange project a comparison was made between several e-business standards and it became clear that the objectives of ebXML were very similar to the objectives of openXchange. The openXchange framework is compatible with ebXML and openXchange contributes to the development of ebXML.

1.2 Business Processes

As said, ebXML offers a language for specifying business processes [Bps02]. This is done in the Business Process Specification Schema (BPSS). The term business process is somewhat misleading since the BPSS does not specify business processes, but business collaborations. Both business processes and business collaborations describe business-related activities that take place in order to achieve a certain business goal (e.g. the placing of an order), and the ordering of these activities in relation to each other. The difference between business processes and business collaborations is that business processes describe the activities from the point of view of only one company. Two types of business processes can be identified: internal and external. Internal business processes describe activities that take place within the company and do not have any interaction with external companies. External business processes also specify interaction with external companies.

When specifying collaborations, the term "party" is often used instead of company. Business collaborations specify the collaboration between parties, by specifying the interaction points of external business processes. In contrast to a business process, a collaboration is not created from the view of one party, and does not focus on that party, but on the collaboration. There are two types of business collaborations: binary collaborations and multiparty collaborations. A binary collaboration describes a

collaboration process between two parties. A multiparty collaboration describes a collaboration between multiple parties.

In ebXML, binary collaborations are described in UML activity diagrams. Multiparty collaborations are specified using activity diagrams, by combining multiple binary collaborations. Activity diagrams offer some modelling flexibility. The same collaboration may be modelled by differently looking activity diagrams. Moreover, if two companies do not support exactly the same collaboration, they might still be able to do business if the collaborations have enough similarities. It is the goal of this M.Sc thesis to develop an algorithm that can match business collaborations on similarity.

1.3 Assignment

This paragraph will state the research goal and problem description and describe the research approach

1.3.1 Title

Matching of ebXML business processes

1.3.2 Research goal

Finding a way to make an automated match between two ebXML business processes

1.3.3 Problem description

In order for two organisations to do business, their business processes must be able to collaborate. Within the ebXML framework, interaction between two organisations is specified with binary collaborations, using UML activity diagrams. A definition of similarity between two collaborations must be given. An algorithm has to be developed and a prototype has to be implemented in order to check whether two binary collaborations fulfil this definition of similarity. Also, the correctness of the solution has to be shown.

1.3.4 Research questions

In order to solve the problem stated in the problem description it has to be split up into a number of sub-problems. In this paragraph each of sub-problems is identified, including the research question necessary to solve them.

ebXML
ebXML offers a framework for b2b e-commerce. This framework includes information about business collaborations. In order to solve the problem stated in the problem description, certain aspect of these business collaborations should be clear

1. *What is ebXML?*
2. *How are business collaborations described in ebXML?*
3. *Define the intended users and the intended use of the algorithm.*
4. *Define when two collaborations are similar.*
5. *Verify the definition of similarity.*

Activity diagrams

Binary collaborations are based on UML activity diagrams. Prior to developing an algorithm for comparing binary collaborations, an algorithm needs to be developed to compare two activity diagrams according to earlier defined similarity.

6. *What are UML activity diagrams?*
7. *Design an algorithm for comparing two activity diagrams according to the definition of similarity*
8. *What methods are available for proving correctness of an algorithm?*
9. *Prove that the algorithm fulfils the definition of similarity*

Binary collaborations
The algorithm for comparing activity diagrams has to be extended in order to compare binary collaborations. To realize this extension, the following questions have to be answered

10. *What elements have to be added to an activity diagram in order to be able to describe a binary collaboration?*
11. *Determine which of these elements are most relevant for the comparison.*
12. *Adjust the algorithm so it can handle binary collaborations*

Prototype
Prototypes are used for different purposes and there are many ways of prototyping. From the start of the project, it was clear that a prototype should be delivered.

13. *What types of prototypes are available?*
14. *Determine the intended use of the prototype.*
15. *Design and implement the prototype.*

Testing
After the algorithm has been developed and the prototype has been build, the algorithm has to be tested.

16. *What methods for testing exist?*
17. *Design the test.*
18. *Test the algorithm*

1.4 Outline of this document

This research focuses on the matching of ebXML business collaborations. In order to successful introduce ebXML in an organisation, people within that organisation must be aware of the advantages of doing electronic business. EbXML, is only a means to achieve a certain goal. Chapter two will start off with pointing out the advantages of doing business in an electronic way, and then introduce the ebXML framework.

Chapter three will introduce the concept of matching, explain why matching is necessary and explain when matching should be used. The problem of matching is twofold. Chapter four will describe a method for the matching of structure of collaborations. A business collaboration (or collaboration), consists of a number of activities. In order to come to a successful matching solution, activities have to be matched on a semantic level. Chapter five will present a solution that can be used to match activities within ebXML.

Finally, some conclusions and recommendations for future work are given in chapter six. Figure 1.2 shows the structure of this document.

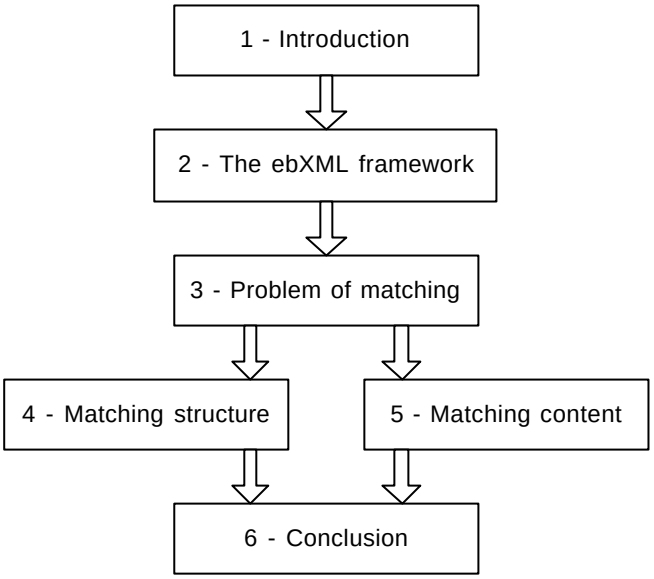


Figure 1.2: Outline of document

2 INTRODUCTION INTO EBXML

This chapter describes several aspects of ebXML. First some background is given on the project and the so-called 'vision' of ebXML will be explained. Once the goal of ebXML is clarified, an introduction to the components of the framework is given. After this introduction, the reader shall be familiar with the most common aspects of ebXML and most of the abbreviations are introduced. Before a company can use ebXML, it has to take some steps. These steps shall be explained to give the reader an impression on what it takes to use ebXML. Finally, after having explained what ebXML is, an impression is given into what ebXML is not. Readers that are familiar with ebXML may skip this chapter and proceed to the problem of matching.

2.1 The ebXML project, the new way of B2B e-commerce?

In 1999, a project was started as a joint effort of OASIS (a non-profit, member based consortium) and the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT, a world-wide organisation that is part of the United Nations). The first project meeting was in November 1999 and 18 month later, in May 2001 some 25 documents were delivered. The intended result of the project is very well caught in one of the first white papers [Whi00] on this subject:

> *"The vision of ebXML is to create a single global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML based messages. ebXML enables anyone, anywhere, to do business with anyone else over the internet."*

To accomplish this vision, a number of working groups has been created, all focussing on one particular piece of the puzzle. Two things were clear from the start: XML and the Internet had to be used. XML offers a lot of flexibility and seemed the ideal standard for exchanging information between different applications. Using the Internet lowers costs since nowadays almost every company is connected to the Internet.

Although the original project was concluded in May 2001, the working groups are still working on the specifications, correcting issues and adding features.UN/CEFACT and OASIS are still involved and more and more companies start using ebXML.

2.2 Using ebXML

This section describes an example that illustrates the steps a company has to take in order to adopt ebXML. Some components of the framework are mentioned to give the reader an idea about the intended use of those components and the relation between components. In the following section, the components are described in more detail.

Consider the following scenario. Cantena is a company that sells food and drinks. Their assortment consists of products that are sold in canteens of sporting clubs and companies. Typical products are instant soup, coffee, candy, fast food and soft drinks. Currently clients fax or phone their orders and these orders are processed manually. Because this manual processing takes a lot of time and is sensitive to errors, Cantena wants to automate the ordering (and invoicing) process. EbXML is chosen instead of EDI since most of Cantena's customers are small or medium sized companies, and these companies do not want to invest is an expensive EDI system. Cantena is a member of an industry organisation, which

already defined ebXML business collaborations and documents for its members and stored these in a public registry.

Multiware is a wholesaler in building and construction materials and offers a large variety of products. Multiware has EDI connection with some of its large suppliers, but the majority of its suppliers do not offer this, simply because EDI is too expensive. Around a year ago, Multiware and some of its suppliers started using ebXML since it offers the advantages of doing electronic business, but the start-up costs are considerably lower than the costs of EDI. EbXML is not only used for communication with suppliers, but also some of the Multiware's clients are using ebXML. Recently some of Multiware's clients (especially building firms) ask for products like coffee, soft drinks, soup and candy bars. Multiware will have to look for a new supplier since none of its current suppliers do not has these kinds of products.

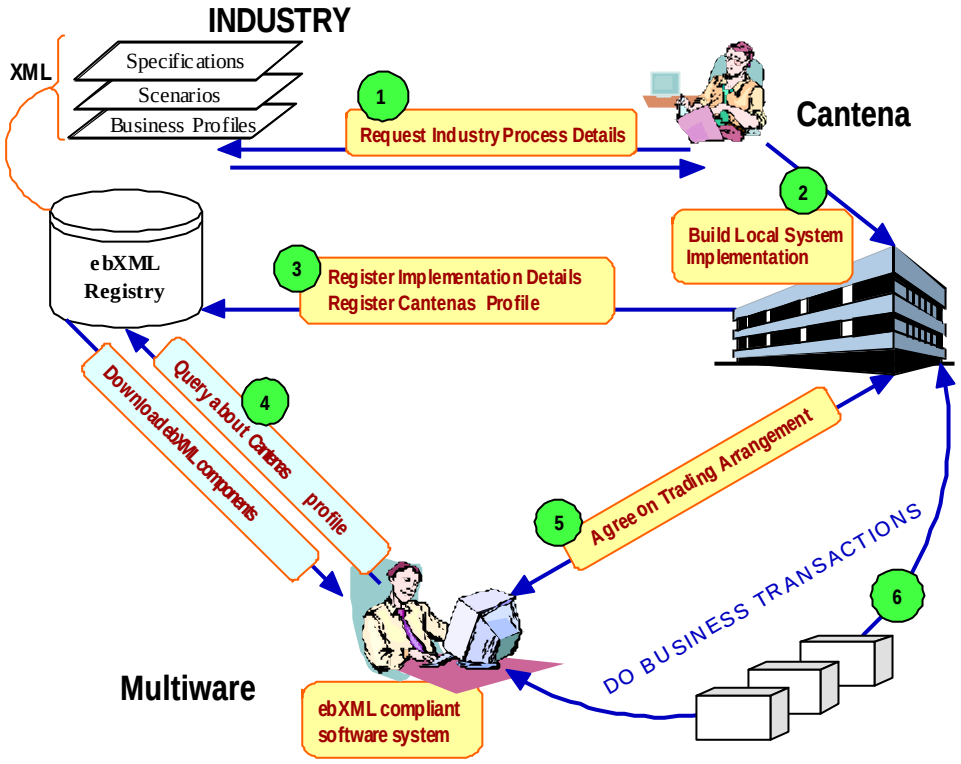


Figure 2.1: business the ebXML way

1. Cantena wants to use ebXML and starts by querying the registry / repository for standardised business collaborations and documents. After searching the registry, Cantena downloads the collaborations and documents provides by the industry organisation

2. Once the choice has been made on which of the business collaborations will be supported through ebXML, the internal systems have to be adjusted. In order to use ebXML, software components have to be build (or bought) that can extract the necessary information from the companies ERP system, and store new information into it. Also a messaging service has to be installed, and choices need to be made about the technical issues (like supported protocols).

3. Once Cantena finished implementing, a profile (called CPP in ebXML) is created that contains a description of the company and references to supported business collaborations. Cantena may choose to use the collaborations and documents that are provided by the industry organisations. If the standard collaborations are not compliant with their internal processes, Cantena may choose to create their own collaborations and reference these from the profile. Once the profile is finished, it is published in a public registry.

4. Multiware is already using ebXML and looking for a supplier that can deliver new products. After doing some searching, Cantena's profile is found and downloaded. Now Multiware has to decide whether or not Cantena is a suitable supplier.

5. Multiware decides that Cantena looks like a suitable supplier, and an agreement (called CPA in ebXML) has to be made on the way they will do business. The profiles of both companies are compared and an agreement is formed. This agreement contains technical issues (like supported protocols) and references to business collaborations and is valid for a specific period. Since Multiware and Cantena both member a different industry organisation, it is likely that they reference different collaborations. This does not necessarily mean they cannot do business. Part of the agreement formation is a matching algorithm that will compare the collaborations.

6. The final step is doing business. In this step, business documents like orders and invoices are exchanged in real-time to support the trade between both companies.

This example shows the steps necessary for a company to use ebXML. The steps 1 to 3 usually have to be done once. Only if a company changes something that is reflected in the CPP (e.g. buy a new messaging system that supports other protocols), it will have to go through step 3 again. Step 4 and 5 only have to take place if a company is looking for a new business partner, or if the agreement expires. The final steps takes place every time a company starts a business collaboration (e.g. every time something is ordered).

2.3 The ebXML framework

In contrast to e.g. EDI, which is only a messaging standard, ebXML is a framework that contains a number of elements that can be used for doing e-business. A company doesn't necessarily have to use all components of the framework, but can adopt just a few of them, depending on its needs.

2.3.1 Component overview

This section gives in introduction into the components of the ebXML framework. Following sections give more detailed information into the components used for matching of collaborations: BPSS and Core Components.

- In a **Business Process Specification Scheme (BPSS)** instance, a company specifies its business processes. This name is somewhat misleading since such a specification only describes those parts of the business process that includes interaction with external parties. Internal activities are omitted. Roughly speaking, a BPSS instance specifies which business documents are exchanged, in which order they are exchanged, and the conditions under which certain activities can take place. A detailed description of bpss can be found in section 2.3.2.

- The **Collaboration Protocol Profile CPP** describes a company's profile considering its e-business capabilities. Besides referencing the supported BPSS files, it also describes technical issues like the transport protocol being uses (http / ftp / smtp / etc). The **Collaboration Protocol Agreement (CPA)** is formed from two CPP's and contains agreements on both technical issues and supported BPSS.

- The **Core Components (CC) and Business Information Entities (BIE)** are in fact the building blocks that can be used to create the business documents used in the BPSS. BIE's are the actual building blocks that are used in a specific context (e.g. the business area of temporary staffing). CC's are the abstract, context free versions of BIE's. Details on Core Components and Business Information Entities can be found in chapter 5. For the time being it is sufficient to know BIE's are small building blocks used for document assembly.

- In ebXML, reusability is one of the key features. The **Registry and Repository** play an important role to achieve this. Industry organisation can publish standard (industry specific) BPSS files and Business Information Entities in the Repository. CPP's are published in the registry and can reference the supported BPSS files in the Repository. Companies are left free though to publish their own BPSS files. Besides the reusability aspect, the registry is used for finding suitable business partners (like the yellow pages).

- The **messaging service** takes care of all communication within ebXML and uses SOAP With Attachments (SWA) as protocol. It used for exchanging messages between two business partners, and for communication with e.g. the registry. SWA uses XML to describe the message format and can use multiple transport protocols like HTTP, FTP or SMTP.

2.3.2 Business Process Specification Schema (BPSS)

Within ebXML the BPSS is used to specify business processes. The BPSS itself is a XML schema, described in [Bps02]. A company can describe its business collaborations as an instance of this schema and publish it in a public registry. This section gives a detailed introduction into the various aspects of the BPSS.

In ebXML, a business collaboration may consists of multiple layers (figure 2.2). A multiparty collaboration describes a collaboration between multiple companies. The example in figure 2.3 describes a multiparty collaboration between 3 companies.

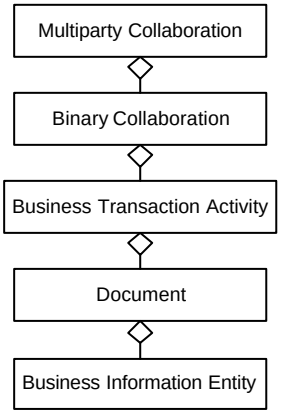


Figure 2.2: Layers in ebXML

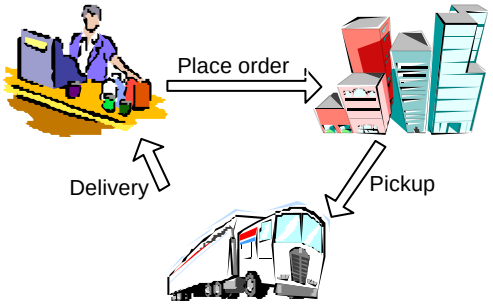


Figure 2.3: Multiparty collaboration

The retailer orders goods from a wholesaler's. A transport company picks up the goods from the wholesaler's and delivers at the retailer. Like all multiparty collaborations in ebXML, the multiparty collaboration in this example is constructed by combining multiple binary collaborations. Every arrow in figure 2.3 represents a binary collaboration.

A binary collaboration consists of a number of activities that are performed by both parties that participate in the collaboration. Binary collaborations are modelled using UML activity diagrams. Each activity in a binary collaboration is either a business transaction (called BusinessTransactionActivity) or a nested binary collaboration. In case of a nested collaboration, entering the activity triggers the start of the nested collaboration. The activity is left immediately after the nested transaction has terminated. An activity can contain pre- and post conditions. The specification is not consistent on the semantics of these conditions. On one hand, the specification states that an activity may not be entered (left) if the pre (post) condition is not valid. This way, the conditions are used as guards on transitions. On the other hand, the specification states that pre- and post conditions do not interfere with the choreography of the collaboration (and therefor only used for effect definition). Besides pre and post conditions, an activity also has a "begins when" and "ends when" expression. According to the specification, an activity immediately starts when the "begins when" expressions becomes true. This is strange though, since it is not guaranteed that the collaboration has already reached the activity (and if pre conditions are used as guards, it is not guaranteed that the guard of the incoming transition is true). The specification does not treat this issue. If the "ends when" condition becomes true, the activity is immediately left.

Another problem using pre conditions, post conditions, begins when and ends when is that the specification gives no language for specifying these conditions. The specification suggest to use OCL (Object Constraint Language), but does not force to use this language, so the user is left free to choice his own language. The lack of a formal language makes it impossible to specify constraints in a clear, unambiguous was.

Each transaction activity consists of the exchange of documents between two parties. Within a transaction activity, there is always one requesting party and one responding party. The requesting party only sends the first document. The responding party may send one or more documents. Every transaction has a couple of parameters including:

- isLegallyBinding (legal status of the documents)
- timeToPerform (maximum time to perform the activity)
- isGuaranteedDeliveryRequired (specifies requirements on messaging service that delivers documents)
- isNonRepudiationRequired (can a party deny sending a specific document)
- isNonRepudiationReceiptRequired (can a party deny that it received a sent document)

Documents can be seen as electronic variants of paper business documents commonly used when doing business. Examples include "order" and "invoice". A document assembled from standardised building blocks called "Business Information Entities", or BIE's for short. The use of standardised building blocks has a lot of advantages concerning reusability and compatibility between companies. The problem however is that, at the moment of writing this rapport, none of the recent specifications describe how this assembly takes place. There is a specification called "document assembly", but this specification is outdated in relation to current specifications.

3 THE PROBLEM OF MATCHING

3.1 Introduction

In the previous chapter, an introduction is made into ebXML. This chapter first explains when business processes should be matched and when a match should be considered successful. Once this is clear, an introduction is given into some of the aspects of matching. Each of the aspects is dealt with in detail in the following chapters.

3.2 Definition of equality

In ebXML, every business collaboration (e.g. the ordering, delivery and payment of goods) consists of a number of activities. Every company has its own guidelines on how, and under what conditions, it wants to participate in a business collaboration. For instance, a supplier of certain goods may decide that it wants to get paid before it will deliver the goods, while another supplier does not care whether it gets paid before or after the delivery. Also the conditions under which a certain activity takes place may vary from company to company. Conditions include:

- pre and post conditions on activities (a temporary worker may only be hired if he has already reached the age of 18)
- legal status of documents (if a document is received, the sender cannot ignore sending it)
- the maximum time that is allowed for an activity (payment must be received within 7 days after delivery).

In ebXML, business collaboration specifications are used to address all these issues and specify how a certain company wants to do business.

Once a company has published its profile (including references to business collaboration specifications) it is ready to start looking for potential business partners. If a new business partner is found, an agreement has to be formed before these two companies can start doing business transactions. This agreement (CPA) includes a specification of the way the two companies shall do business. It seems evident that a collaboration specification included in a CPA needs to be supported by both companies. To determine this, the collaboration specifications of both companies need to be compared. Currently ebXML does not offer any functionality to do this in an automated way, therefore the need arose within the openXchange project to create a system that can do this.

If two companies reference the same business collaboration, for instance one that was created by the branch organisation, and both companies take a different role (e.g. buyer and seller), we speak of a trivial match. The collaboration specification that both companies have in common can be referenced from the CPA and the two companies are ready to start doing business.

Sometimes however, companies may reference different collaborations. This does not necessarily have to mean that these companies cannot do business together. In some cases, a new collaboration can be formed that is supported by both companies. This collaboration that should be stored in a new bpss document (called bpss ‘a’ in figure 3.1), describes the compatible parts of two collaborations and can be considered as the “agreement” of two collaborations. It is the task of the matching algorithm to form this ‘agreement’ collaboration and store it in a bpss document. Before an algorithm can be created it has to be

clear under what conditions (parts of) two collaborations can be considered compatible, or to be more precise, what the definition of a successful match is.

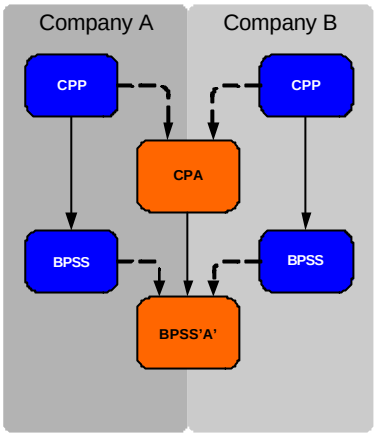


Figure 3.1: Agreement on BPSS

This research only focuses on the matching of business collaborations. Automated generation of a CPA is also a focus within openXchange, but is not addressed in this research. The main goal for the matching algorithm is to determine whether there is a way for the two companies to do business together. One can consider a collaboration as a specification of all possible business scenarios that are supported by a company (for that specific business process). This brings us to the following definition: *Two business collaborations match if there is at least one successful business scenario that is supported by both process specifications*. It is hard (if not impossible) to formally that this is a useful definition, but it was checked and agreed upon by several experts on this subject. Therefore this definition is used as the starting point when developing the algorithm.

3.3 The two aspects of matching

The problem of matching is twofold. First the structure of two collaborations need to be compared. As mentioned earlier, a business scenario consists of a number of activities that need to be done. A collaboration specification defines all possible orders of these activities. In some cases there may be some differences in this ordering, but still there might be some orders that are supported by both specifications

Consider a simple example to illustrate this (figure 3.2). Suppose that two companies reference different collaboration in which two activities take place: the delivery of goods and the payment of these goods. The supplier first wants to receive his payment before he is going to deliver. The consumer on the contrary has no preference. Even though it seems obvious that these two companies can do business together by first paying and then delivering the goods, standard ebXML offers no functionality to check this compatibility. The algorithm has to detect and specify this collaboration.

The second part of the matching is the matching of content or semantics. In the previous example there were two activities, delivery and payment. The assumption is made that the delivery activities of the buyer and the seller are equal, but this doesn't necessarily have to be the case. Sometime companies use the same name but actually specify different activities or specify different conditions. Also companies may use different names, but actually specify the same activity. To solve this problem, the algorithm has to check whether or not the activities are equal. If two activities are different, the algorithm has to check whether or not these differences can overcome.

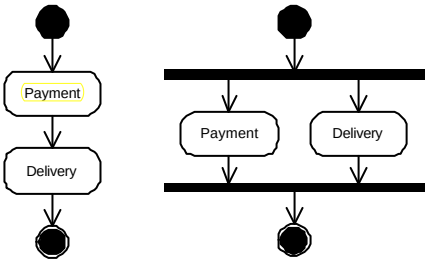


Figure3.2: Differences in choreography

In the next chapter the matching of structure will be explained. In chapter 5, the matching of content shall be addressed.

4 MATCHING STRUCTURE

4.1 Introduction

Since Multiparty Collaborations are top-level process specifications, it seems logical to start the matching algorithm at this type of collaboration. However, for the following reasons the choice is made to start matching on the level of binary collaboration:

- The matching algorithm is meant to be used as part of an algorithm for automated CPA generation. A CPA is an agreement between two parties and references binary collaborations.

- A company is only interested in business transactions it participates in. Third party collaborations are not that interesting. Related to this is the fact that in some complex real-world business scenarios, none of the participating companies may have a complete view on the total collaboration.

- The specifications are still under development, and they still contain some errors. The bpss specification, which describes both binary and multiparty collaborations, especially has some flaws on the specification of the multiparty collaborations. It is unclear how to link companies to a specific role, and more companies can be linked to the same role.

4.2 UML Activity diagrams

In ebXML, binary collaborations are expressed in UML activity diagrams. In order to build an algorithm that can match binary collaborations, first an algorithm has to be created that can match activity diagrams. This paragraph will describe the UML activity diagram notation style. It is intended to give the reader an impression on this notation style and will not go into all formal aspects. Only those aspects necessary for this thesis will be treaded.

An activity diagram is a directed graph, consisting of nodes and directed edges An activity diagram describes a system. Nodes describe the possible states of the system while edges describe the allowed state transitions. The UML activity diagram notation style defines one type of edge and several types of nodes. The various types of nodes are shown in figure 4.1. First the semantics of all possible nodes are treated and then some of the problems concerning the semantics are addressed.

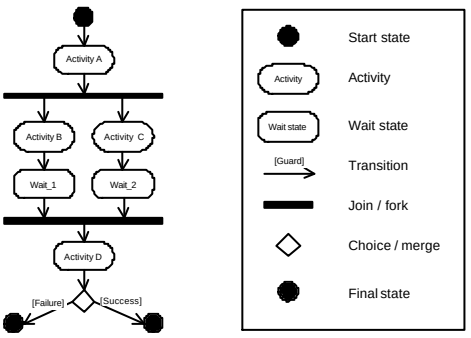


Figure 4.1: UML Activity Diagrams

In the activity diagram notation style, edges are called transitions. A transition always has one source node and one target node, and may contain a guard. If a transition contains a guard, it can only be taken if the guard is valid. The system can never be "in a transition, a transition from one node to another does not consume any time. This means the diagram is always in one (or more) nodes. An activity diagram is

deterministic, this means that, unless guards are used, each node can only contain one outgoing transition (e.g. activity D in figure 4.1). The only exception to this rule is the fork node which does have multiple outgoing transitions. To preserve determinism when a node has more outgoing transitions, guards have to exclude each other (formally, the UML does allow guards to overlap, but for this research we do not).

Within activity diagrams, two categories of nodes can be identified. First, there are the 'normal' nodes, which are used to describe the state of a system. The other category consists of 'pseudo states'. The system can never be in a pseudo state and, just like a transition, a pseudo state doesn't consume any time. Pseudo states therefore are syntactic sugar, used to glue edges together.

The first pseudo state is the start state. Every diagram may only contain exactly one start state. This start state has exactly one outgoing transition and no incoming. The outgoing transition of a start state may not contain a guard. A system stops once it is in a final state. The final state only has incoming transitions, never an outgoing transition. Incoming transitions of a final state may contain a guard.

A choice state has one incoming transaction and multiple outgoing transaction. The incoming transition of a choice state may not have a guard, all outgoing transactions must contain a guard. As explained, the guards on transitions leaving a choice state have to exclude each other. Besides the fact that guards must exclude each other, always one guard must be true (or else the system could get stuck in a choice node). One way to make sure that one of the guards is true is by adding a [else] guard. This [else] guard is true if all other guards are false. A merge state allows multiple incoming transactions and has one outgoing transaction.

The last pseudo states are the fork and join. When a fork state is used, the thread splits up into several parallel threads. Every fork state has exactly one join state. In a join state, parallel threads are merged. Before parallel threads can be merged, all threads must have reached the join state, meaning all states that have an outgoing transition leading to the fork must be terminated.

The UML activity diagram notation style identifies three types of normal states: action, wait and composite states. When the system enters an action state, the execution of an atomic activity is triggered. The system will leave the state immediately after the activity is finished. The activity is atomic, meaning that either the complete activity is executed or the activity is not executed at all. The atomic activity cannot be interrupted. In a wait state, the system waits for external event to leave the state. An external event can either be an event that happens outside the system, or a state transition within the diagram. No activities are performed in a wait state. In a composite state, another, nested activity diagram is executed. The execution of the nested diagram starts once the composite state is entered. The composite state is left once the nested diagram terminates.

UML activity diagrams lack a formal semantics. Some of the problems that arise because of this lack are treaded in [Esh02]. One of the issues relevant for matching of activity diagrams is the termination of activities that lead to a fork state. Consider the example in figure 4.2-a (which is valid according to the UML specification). Both activities A and B are action states, meaning that the outgoing transitions are triggered by internal activities. Suppose activity A terminates while activity B is still running. Termination of A means that the system cannot stay in action state A so the outgoing transition must be taken. This causes a problem since all transitions leading to a join must be taken at the same time. Action state may not be left yet since the atomic activity B is not finished. To avoid this problem, transitions to a join may never originate from an action or composite state, but only from a wait state as in figure 4.2-b.

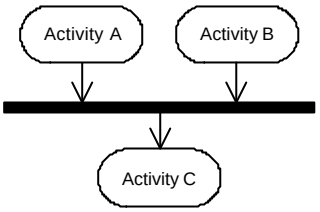


Figure 4.2a: Join w/o wait state

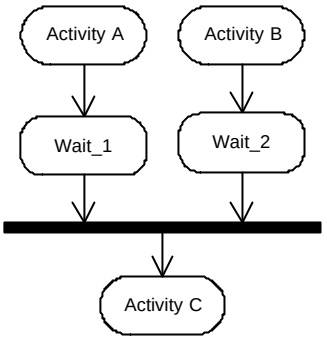


Figure 4.2b: Join with wait state

4.3 The problem of matching activity diagrams

Now the introduction to process specifications and activity diagrams is made, the first step in process matching can be made. The criteria for a successful match will be given first. There are currently no techniques for matching activity diagrams, so some older techniques were examined whether or not they would apply in a new setting.

4.3.1 How to match?

Before an algorithm can be created to match two activity diagrams, it has to be clear when two activity diagrams are considered to be 'compatible'. This means a definition has to be created for a successful match. The definition of equality for business processes (page 17) is *"..there must be at least one business scenario that is supported by both"*. Since each business scenario is a run of activities through the activity diagram, the algorithm should give a successful match if there exists at least one run that can be taken in both activity diagrams.

Preferably, the algorithm does not only check whether or not there is such a run (and give that run), but also give all possible runs that can be taken in both diagrams. This set of possible runs can itself be described as an activity diagram. To conclude, the algorithm takes two activity diagrams as input and, if there is a successful match, create a new activity diagrams that represents all possible runs.

4.3.2 Techniques used

Although a lot of research has been done on activity diagrams, there are no techniques available to compare two diagrams. Therefore research has been done on a number of older techniques, in order to check whether they would be usable for the comparison of two activity diagrams. The most obvious candidate is bisimulation, as used in process algebra. Another possible solutions seemed to be using set theory in order to determine trace equivalence. These two techniques, and the problems identified when using them, will be treated. Some other techniques that were tried, like the use of logic, will not be dealt with in this report.

Simulation

In ebXML, activity diagrams are used to represent a business process. A diagram technique called process graphs as used in process algebra has a lot in common with activity diagrams. A process graph consists of nodes (that represent atomic actions) and of transitions between these nodes. In process algebra, a technique called bisimulation is used to define an equivalence relation between two process graphs. If there is a bisimulation relation between two process graphs, every trace in one graph is possible in the other graph and vice versa. This relation is too strong for the goal of matching business processes within ebXML since we only need one trace that exists in both graphs.

If the bisimulation is weakened in such a way that every trace of one graph is possible in another graph (but not vice versa), the relation is usable. We call this relation a simulation relation. Consider two graphs: graph A and graph B. Graph A is simulated by graph A' if:

1. Every node *a* in graph *A* has an equivalent node *a'* in graph *A'*.
2. For every edge (*a*,*b*) in graph *A* there is an edge (*a'*,*b'*) in graph *A'*.

How can this simulation technique be used to compare activity diagrams? The goal of the algorithm is to find all possible traces possible in two given activity diagrams A and B, and represent these traces as a new diagram C. According to the previous definition, C is simulated by both A and B.

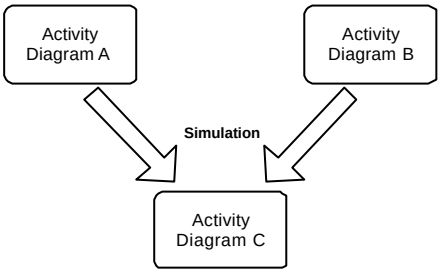


Figure 4.2: Simulation

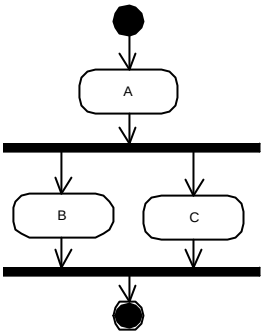


Figure 4.3: Parallelism

Although there is a lot of resemblance between activity diagrams and process graphs, there is one big difference: activity diagrams can contain parallelism, introduced by a fork node (see figure 4.3). Without parallelism, the state of a system is always represented by exactly one node. With the introduction of parallelism, the state of the system is represented by multiple nodes at the same time. This makes the simulation relation unusable. There are some variations on process algebra that also support parallelism, but this is another form of parallelism. More details on the differences in parallelism are treated in section 4.4.1

Set theory

Another technique examined is set theory. The goal of the matching of two activity diagrams is to determine whether there exists a trace that can be performed in both activity diagrams. If an activity diagram is converted into a set of all possible traces, two diagrams A and B can be matched by checking whether there is overlap in the set of possible traces of both diagrams. This overlap represents the (partial) trace equivalence of A and B and can be represented by an activity diagram C. Figure 4.4 illustrates this.

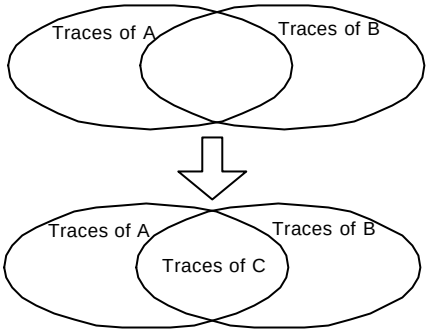


Figure 4.4: Set theory

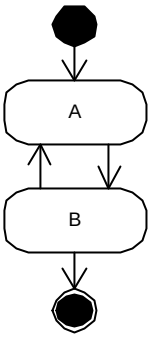


Figure 4.5: Problem with cycles

As with the simulation solution, a problem is encountered when applying the set theory. The algorithm requires calculating all possible traces. In large activity diagrams this can become a problem, since it required calculation power grows exponentially with the size of the diagram. In some cases, where the diagram contains a cycle (figure 4.5), it is impossible to calculate all possible traces since the number is infinite.

4.4 Towards a solution

All efforts to apply older, well-known techniques to solve the matching problem failed. In all cases this was either because of parallelism or because of cycles. One way of solving this is to make the assumption (and specify it as requirement) that one of these constructs cannot be used. This would of course mean that the algorithm isn't generic anymore. Another possible solution would be to eliminate one, thereby keeping the solution generic. One way to eliminate parallelism is explained in [Pra91].

4.4.1 Eliminating true parallelism

Before eliminating parallelism its important to understand that two types of parallelism can be identified. First there is true parallelism in which activities take place exactly at the same time and may influence each other. The other form of parallelism is branching time parallelism. In branching time parallelism, all activities are actually executed sequentially, but the order in which the execution takes place is undefined (so actually, this is a form of non-determinism).

Consider the example in figure 4.6 where two activities are specified to execute parallel (a). There are three possible execution scenarios (b): first A then B, first B then A or A and B simultaneously. If only the first two scenarios would be possible, the parallelism would be branching time and the diagram could be transformed relatively easily into a sequential one (c). The third scenario causes problems, since it's not clear what state changed the system undertakes before the final state is reached. This is a case of true parallelism.

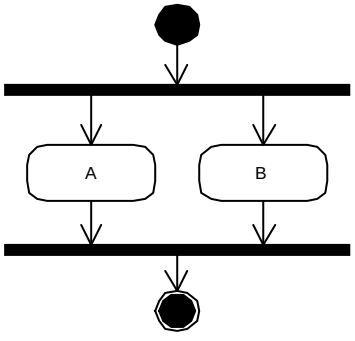


Figure 4.6(a)

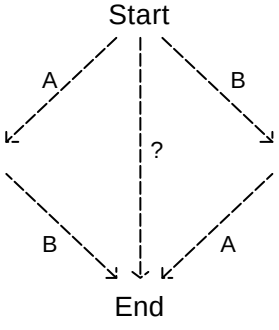


Figure 4.6(b)

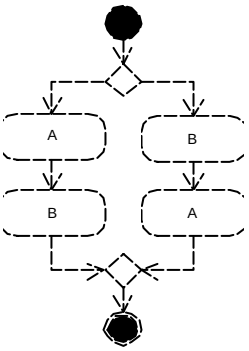


Figure 4.6(c)

The problem of true parallelism could be illustrated by the following example. Consider the parallelism in figure 4.6(a) to be true parallelism. Suppose if activity A could be decomposed in the sequential execution of u,v and w while B can be decomposed in x,y and z. In this case, the diagram in figure 4.6(c) does not have the same traces as the diagram in figure 4.6(a). Diagram (c) only supports two executions: u,v,w,x,y,z and x,y,z,u,v,w while diagram (a) also supports e.g. x,u,y,v,z,w. If the parallelism in figure 4.6(a) is considered branching time parallelism, then both diagrams do have the same possible traces.

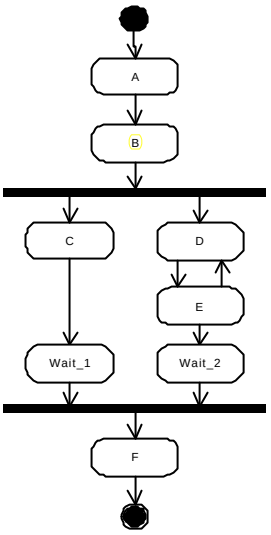


Figure 4.7: Trace history

According to [Pra91], a diagram using true parallelism can be considered as a diagram using branching time parallelism, as long as all states (or activities) are atomic. To achieve this, every nested activity (in the previous example both A and B) should be expanded in order to remove the nesting.

4.4.2 State transition systems

One of the mayor problems in comparing activity diagrams is the use of parallelism (either true or branching time). Once parallelism is introduced, the state of a system is no longer represented by one node (or activity), but by multiple nodes. This eliminates the possibility of using simulation to compare two diagrams. After converting true parallelism into branching time parallelism, the activity diagram can be transformed into a standard State Transition System (or STS) containing no parallelism. In a STS, each possible state of the system is represented by exactly one node.

If an activity diagram is to be represented by a STS, each possible combination of states in the activity diagram must be represented by exactly one node in the STS (e.g. CD and CE). This implies that the STS representation will have at least the same amount of states. The number of states grows with the number of branches and the number of states in each branch. Given a partial STS *S* representing a parallel part *P* with *n* parallel branches of an activity diagram, the maximum number of states of *S* can be calculated as:

- $\max(\#S) = \prod_{x=0}^{n} (\#P_x)$

where #X is the number of states in X and P_n is branch number n.

In an activity diagram, a state transition is made immediately after an activity terminates, therefore termination of an activity must lead to a transition in the STS. Suppose the activity diagram is in two states and both states have one outgoing transition (e.g. C and D in figure 4.7). The STS representation has one state (CD) and two outgoing transitions. After termination of one of the two activities, it is clear what the next state of the activity diagram will be (termination of D leads to states C and E while termination of C leads to Wait_1 and D). To make this distinction in the STS (which has two outgoing transitions), each transition needs a guard stating the termination of one of the activities. Note that simultaneous termination of both activities is not possible if both activities are atomic.

State Transition Systems are either deterministic or non-deterministic. In a deterministic system, the state of the system can always be determined given the trace history. In a non-deterministic system this is not always the case. Consider state E in figure 4.7: after termination of E, either Wait_2 *or* D is active (and of course C or Wait_1), but it is not clear which one. Although it is possible to create a deterministic STS of each activity diagram, an algorithm for creating a non-deterministic STS is more strait forward.

According to [Sud97], every non-deterministic STS can be transformed into a equivalent deterministic STS. For the purpose of matching, a deterministic STS is preferred.

4.4.3 Matching two activity diagrams

The first step in matching two activity diagrams is converting each activity diagram into an activity diagram in which each activity is atomic. This is done by inspecting each activity and splitting it up into several activities. No generic algorithm can be presented since atomicy depends on the nature of the activities. After this conversion, the both activity diagrams are converted into a non deterministic STS using the algorithm presented below.

Wait-states were introduced to make a correct synchronisation in a join node. No activity is done in a wait-node and therefore wait-nodes are not relevant for matching. Moreover, since wait-nodes are only introduced before a join node, an activity diagram without parallelism shall never contain a wait node, meaning that an activity diagram that contains parallelism shall never successfully match a diagram without parallelism. This is unwanted behaviour and therefore wait states are removed from states in the STS. In an activity diagram, just before a join, the state of the system is represented only by wait states (e.g. in figure 4.7, the state of the system *before* state F is (Wait_1,Wait_2). Removing all wait states from states in a STS means that for every join in the activity diagram, the STS contains an empty state. This empty node shall be removed and all incoming transitions shall be connected to the destination of the outgoing transition. The example in section 4.4.4 demonstrates this.

After creating a non deterministic STS, a deterministic STS is created for each activity diagram according to transformation rules explained in [Sud97]. The match of the two original activity diagrams can be calculated by generating the intersection of two deterministic STS diagrams. This intersection is represented as a third STS which can be transformed back into an activity diagram.

Algorithm 1 uses functions the following functions that will not be specified in detail:

- ***successor(AD,N)***. This function returns the nodes (activities) in activity diagram *AD* that can be reached from node *N* by taking exactly one transition. The result of this function is:
  - and({node}) if a single node or a fork node is reached after taking the transition. If a fork node is reached, {node} contain the nodes that are directly reachable from the fork. In the activity diagram AD depicted in figure 4.7, the function *successor(AD,D)* will return *and({E})*, while *successor(AD,B)* will return *and({C,D})*
  - or({node}) if *N* has more than one outgoing transition, or if *N* has a transition that leads to a choice node. In case of a choice node, *{node}* will contain the set of nodes that are reachable from the choice node. In figure 4.7, *successor(AD,E)* will return *or({Wait_2,D})*
  - join({node}) if the outgoing transition of *N* leads to a join node. {node} contains the set of nodes that also have a transition leading to *N* (these nodes should all be Wait nodes according to paragraph 4.2). Note that this is different from the previous two results where {node} contains the set of nodes that are reachable from *N*. In figure 4.7, *successor(Wait_1)* returns *join({Wait_1,Wait_2)*.
- ***getNode(AD,N)*** returns node *N* from activity diagram *AD*
- ***hasNode(GR,N)*** returns true if graph *GR* contains node *N*. Otherwise, this function returns false.
- ***hasEdge(GR,O,D)*** returns true is graph *GR* has an edge originating from node *O* and leading to node *D*. Otherwise, this function returns false.
- ***addNode(GR,N)*** adds node *N* to graph *GR*
- ***addEdge(GR,O,D,L)*** adds an edge in graph *GR* from node *O* to node *D* with label *L*

Within the algorithm, four blocks can be identified. These blocks are numbered and treated after the algorithm is given.

Algorithm 1

```
ADN :: activity                                                                  1
GRN :: {activity}+                                                               1
ADE :: (source :: ADN,dest :: ADN)                                               1
GRE :: (source :: GRN,dest :: GRN,label::string)                                 1
STS :: ({GRN},{GRE})                                                             1
ActivityDiagram = ({ADN},{ADE})                                                  1

Graph = new(STS)                                                                 2
ActD = read(ActivityDiagram)                                                     2

newNode :: GRN                                                                   2
newNode = getNode(ActD,start)                                                    2
addNode(Graph,newNode)                                                           2
process(Graph,newNode,ActD)                                                      2
end                                                                              2


Function process(GR::STS, GrNode::GRN, AD:: ActivityDiagram)                     3
newnode :: GRN                                                                   3
FOR EACH a ∈ GrNode DO                                                           3
        succ = successor(getNode(AD,a))                                         3
        IF succ = join(XS) THEN                                                  3
                IF XS ⊆ GrNode THEN                                              3 A
                        newNode = GrNode – XS ∪ successor(joinNode)              3 A
                        addAndProcess(GR, GrNode, newNode)                       3
                ELSE                                                             3 B
                END IF                                                           3 B

        ELSE IF succ = and(XS) THEN                                              3
                newNode = (GrNode - a) ∪ succ                                    3 C
                addAndProcess(newNode,a)                                         3 C

        ELSE  (**succ = or(XS) **)                                               3 C
                FOR EACH x ∈ XS DO                                               3 C
                        newNode = (GrNode - a) ∪ x                               3 C
                        addAndProcess(GR, GrNode, newNode, x, AD)                3
                END FOR                                                          3
END process                                                                      4 A

Function addAndprocess(GR::STS, GrNode::GRN, newNode::GRN, label::string, AD::ActivityDiagram)   4 B
IF hasNode(GR, newNode) AND hasEdge (GR, GrNode,newnode) THEN                    4
        (** do nothing **)                                                       4 C
ELSE IF hasNode(GR, newNode) THEN                                                4 C
        addEdge(GR, GrNode, newNode, label)                                      4 C
ELSE                                                                             4
        addNode(GR, newNode)                                                     4
```

The algorithm starts with type definitions (1) which lead to the types STS and ActivityDiagram. In block (2), an empty STS is created and an activity diagram is read as input. The algorithm will eventually add nodes and edges to the empty STS thereby creating a different representation of the input activity diagram. The first step is to create a start node in the STS. This start node is processed by calling the recursive procedure *process* (3). If a node is processed, the procedure first checks what states are reachable in the activity diagram from the node it processes by calling the procedure *successor*. Dependent on the outcome of this procedure, the algorithm chooses one out of three options (3A, 3B or 3C)

If the successor function returns an and-value or a join-value (block 3A and 3B), the algorithm creates one new node that is to be added to the STS. In case of an or-value, the algorithm creates several nodes that are potentially added to the STS. Each node represents a state of the system represented by the activity diagram. If a new node is created, the function addAndProcess is called (4). This function first checks if the new node already exists and, if so, if there is exists a transition between the node it is currently processing and the newly created node.

If both the node and the transition exist, the algorithm does nothing with the new node and does not call itself recursively (4A). If the new node already exists, but there is no transition between the currently processed node and the new node, this transition is added. As in the previous case, the algorithm does not call itself (4B). If both the transition and the node do not exist in the STS, they are both added and the algorithm continues by processing the new node (4C)

****

When comparing two activity diagrams, the step is to transform both diagrams into a STS using algorithm 1. Then all Wait states have to be removed (for reasons mentioned earlier). Once this is done, the two newly formed (non deterministic) STS diagrams can be transformed into deterministic STS diagrams. This can be done by using algorithm 2.

Algorithm 2

Stap 1
Stap 2

For the purpose of business process matching, this match does not have to be an exact match. The intended result of the algorithm is to check whether or not there is a trace that exists in both diagrams. In order to detect this, a third STS is created representing the intersection of two deterministic STS diagrams.

Algorithm 3

Stap 1
Stap 2

Algorithm 3 creates a STS (C) that is simulated by both original STS diagrams. According to the definition of simulation, every trace possible in STS C is also possible in both STS A and B. We speak of a successful match if STS C contains a trace that leads to an End-node. A third algorithm can be used to

convert the STS back into an activity diagram. The algorithm that is described below is a simplistic one that does not introduce parallelism. Although it is usable, a new algorithm should be designed that can reintroduce parallelism. The example in the following chapter assumes that such an algorithm is used.

Stap 1
Stap 2

4.4.4 Example

In the following example, two activity diagrams, A and B are being compared. This comparison will lead to a third activity diagram C that represents the matching of A and B. Both diagrams are shown in figure 4.8 and contain the same activities (or nodes), but have a different choreography. All activities in both A and B are atomic. The first step is to translate the activity diagrams into STS, according to algorithm 1. The result of the transformation is shown as STS A and B in figure 4.9. Removing the wait states results in STS A' and STS B'. Once the two graphs are created, the second algorithm can be used to create the third graph. Figure 4.10 shows STS C that is simulated by both STS A and STS B, and the activity diagram that is created after converting the STS back into an activity diagram. No formal proof shall be given here for the fact that this is a corrected simulation relation. A graphical representation of the simulation relation is shown in appendix B.

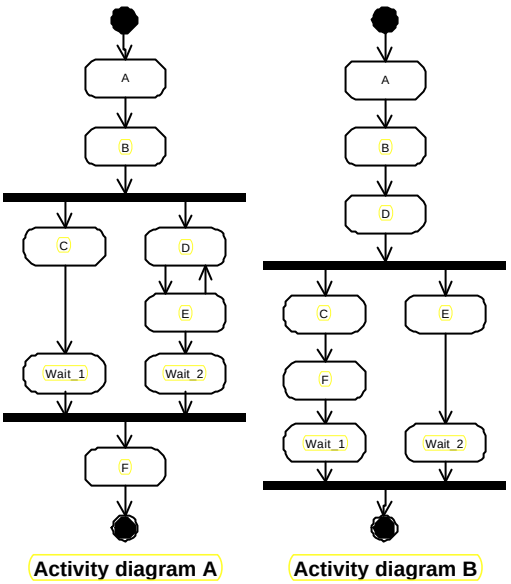


Figure 4.8: Example activity diagram

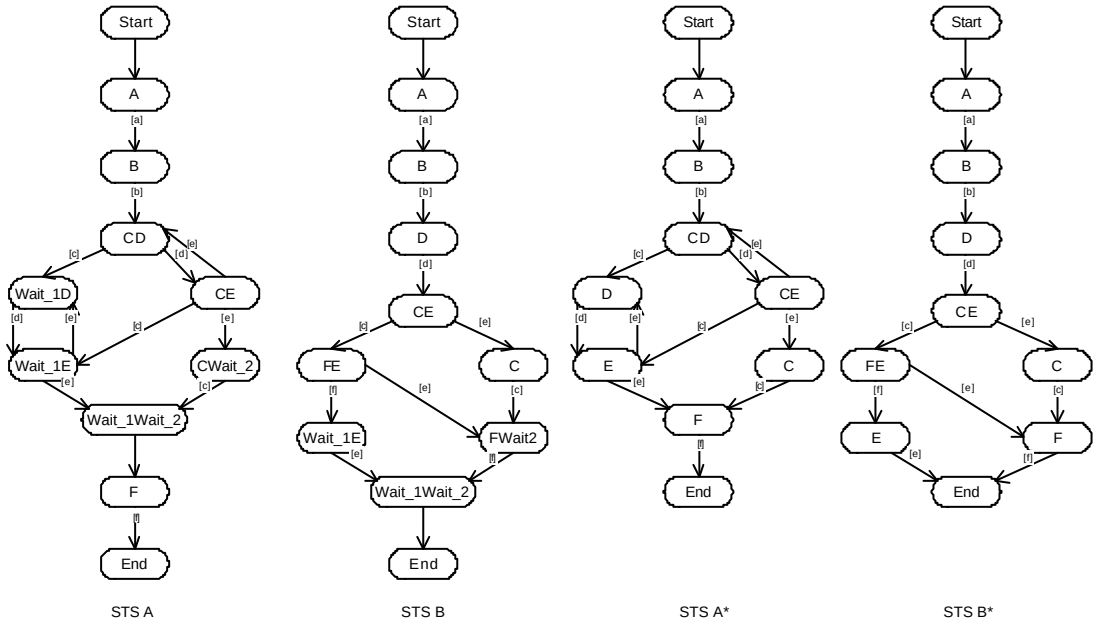


Figure 4.9: Resulting STS

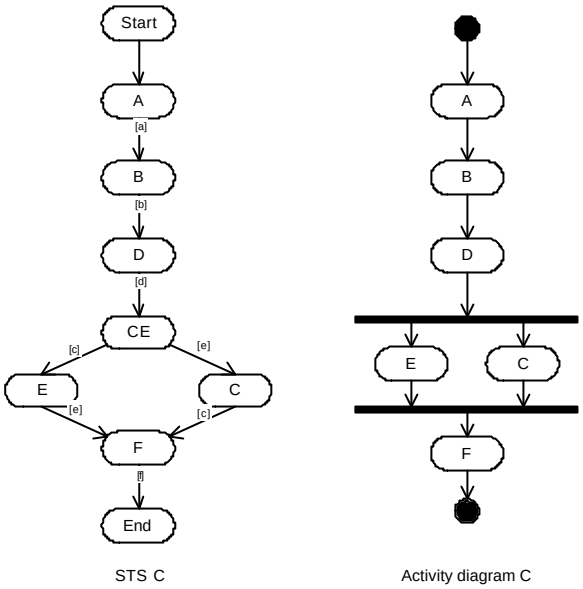


Figure 4.10: Result of matching

4.4.5 Transformation to ebXML

In ebXML, binary collaborations expressed in UML activity diagrams. The algorithm designed for matching activity diagrams can therefore be used to match binary collaboration. The solution described in this chapter can be used to match to structure of collaborations, but in order to come to an useful match, more has to be taken into consideration.

In ebXML, activities in a Binary collaboration are either nested collaborations or Business Transaction Activities. Neither of these two are atomic, so in order to use the algorithm, the following conversions have to be done:

- For each activity A that contains a nested collaboration, the nesting must be removed. This is done by connection the all incoming transitions of A to the first normal node in the nested diagram. Each transition in the nested diagram that leads to an end state should be connected to all outgoing transitions of A.
- If a Business Transaction Activity (BTA) consists of the exchange of more than one document, that BTA must be replace by a sequence of BTA's, each consisting of the exchange of exactly one document. If the replace BTA contains pre conditions, these pre conditions are added to the first BTA in the sequence. If the BTA contains post conditions, these post conditions are added to the last BTA in the sequence.

Besides the matching of structure, also content has to be matched. The following chapter treats the matching of content in ebXML.

5 MATCHING OF CONTENT

5.1 Introduction

In order to come to a successful match of business processes, not only structure but also content needs to be matched. The matching of content consists Several parts. This chapter will start with explaining the role of documents within ebXML. Once this is clear, the matching of documents is treated. EbXML uses small building blocks (Core Components and Business Information Entities) to construct the documents. The matching does not only match syntax, buy also the context in which these blocks are being used. This chapter ends with treating the problem matching conditions.

5.2 Documents in ebXML

5.2.1 The purpose of document exchange

Within ebXML, a collaboration describes a number of activities in which your company is collaboration with another company. In this collaboration both companies have to take some actions. When your company e.g. is going to order something from a supplier, the supplier needs to be informed about this. This seems trivial but it describes the essence of document exchange, since documents (and signals) are the only way to communicate between business partners. So, the purpose of document exchange is to synchronise the view both parties have on the collaboration. Of course, this synchronisation may have legal consequences so the document has a legal status as well.

There are a lot of initiatives involving electronic document exchange, going way back to the 80's with EDI, but also recent XML based messaging standards like hrXML (Human Resource XML [Hrxml]) and UBL (Universal Business Language [Ubl03]) describe electronic document formats. So, what is so different about ebXML? One of the mayor drawback of e.g. EDI is that agreements have to be made with each of the business partners on what document format is going to be used. Once agreed upon, this format is fixed. Other initiatives, like UBL, describe standard documents (e.g. invoice) that offer little flexibility. In UBL, users are not able to create their own document definition.

In ebXML, small, standardised building blocks are used to construct documents. If a user needs a building block that does not yet exist, the user is left free to design his building block and submit it for standardisation. In the ideal case, branch organisations specify standard building blocks for their specific industry, and members may use (or alter) these documents. This way, the users have maximum flexibility and standard components can be defined once and used many times.

There is only one drawback on this flexibility. It could very well happen that two potential business partners both use different business documents. In such a case it is not always clear whether or not these companies are able to do business. Every time a company sends a document, that company's view on the status of the collaboration has changed. The purpose of sending a document is to update the view the other company has on the collaboration. So, in order to check whether two companies can do business, the algorithm must check if the information need of the receiving party is satisfied by the senders document, for each time a document is exchanged.

5.2.2 Core Components and Business Information Entities

Core components (CC's) and Business Information Entities (BIE's) are the building blocks used for creating documents. Essential for all these building blocks is that, besides syntax, they have semantics. This definition may be a little vague but it will become clear later on. Just to give an illustration: a date does not have semantics, it just specifies a certain point in time. Your birth date does have semantics, it means something to you, its more than just a date. An invoice date is also a date, just like a birth date, but it means something completely different. The difference between these two dates is in the fact that they have different semantics and therefore comparing an invoice date with a birth date usually makes no sense.

There are three variants of building blocks: Basic, aggregated and associated. Basic building blocks are elementary parts that cannot be split up into smaller parts. An example of a basic building block is e.g. street name. An aggregated building block consists of one or more properties. Each property is either a basic building block or an associated building block. An association building block is a reference to another aggregated building block. By using these associated building blocks, relations can be created. Figure 5.1 shows an example to illustrate the relation between these three variants.

Figure 5.1: Core Components

The example in figure 5.1 shows the following core components:

- Person (Aggregated)
- Name (Basic)
- Birth Date (Basic)
- Official Address (Association)
- Address (Aggregated)
- Street (Basic)
- ZIP Code (Basic)
- Town (Basic)

Even though the specifications [Cct02] have a clear definition of the difference between a CC and a BIE, in practice it is sometimes hard to use this definition. According to the specification, the difference between a BIE and a CC is that the latter does not have a context and the first does (see also figure 5.2). A number of context categories are specified, including regional and industry context. A BIE is always based on a CC and has some context added. Context is specified by using one (or more) context categories. The concept of using context is a very powerful mechanism that turns out to be the key to the matching of documents, but more on that later on.

Every basic building block (both BCC, Basic Core Component and BBIE, Basis Business Information Entity) has a certain data type. This data type specifies the range of possible values valid for that specific component. Typical data types are integer and string. Every data type is based on a core component type (CCT). There are 10 CCT's which are the basic types of ebXML. Examples are amount, text and Date

Time. A BCC or BBIE can never be based directly on a CCT, only on a data type. Figure 5.2 describes the relation between the different parts of ebXML[1] used in message assembly.

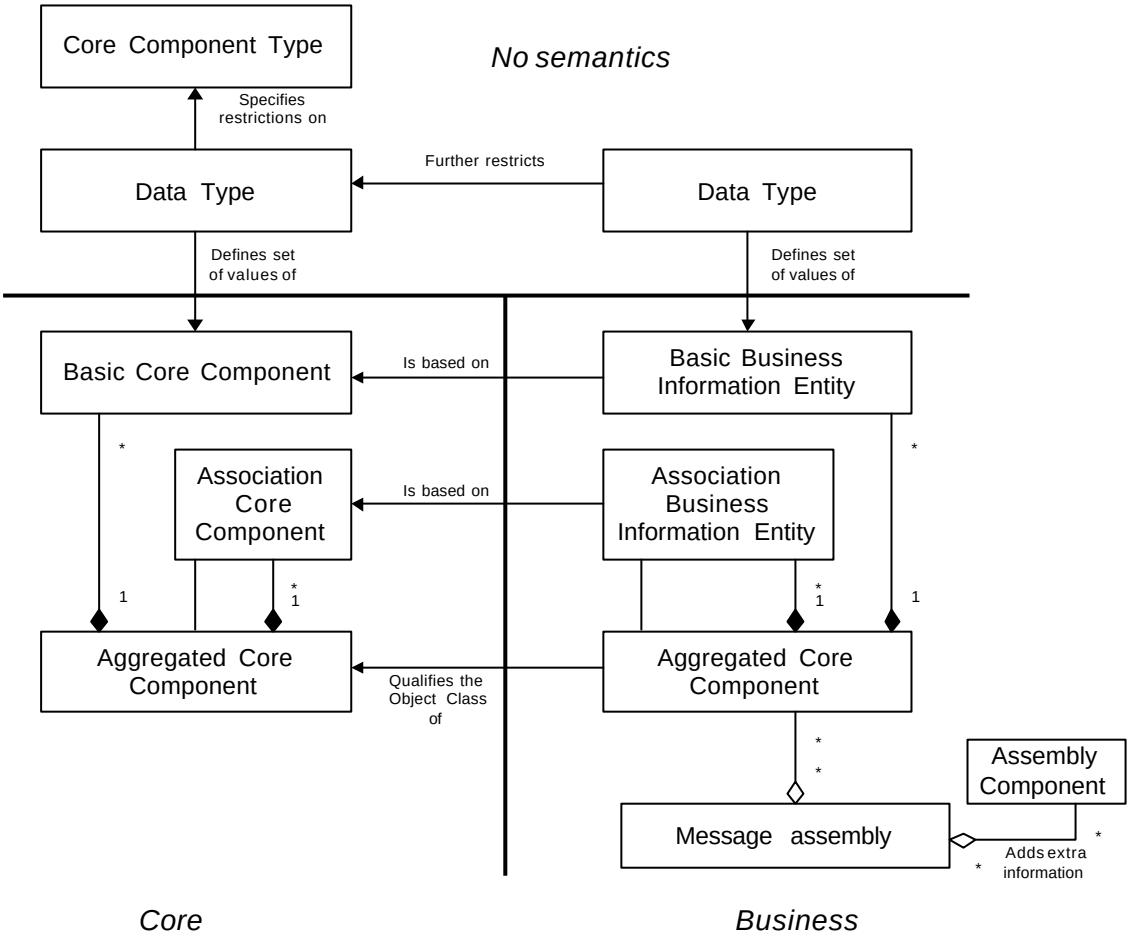


Figure 5.2: Semantics and Context (adapted from [ccts])

5.2.3 Use of qualifiers

A Business Information Entity contains a lot of meta data on that particular piece of information it describes. This information contains a lot of written text like description, all context area's, etc. For the standardization process, a worksheet has been created in which all this information can be filled and submitted to the standardization organisation. All this information would cause a enormous overhead if it were sent with every business transaction. Obviously, this is not desirable. An example of such a worksheet is given in appendix C.

The name of each Core Component is unique. If a mechanism would exists in which the name of every BIE is unique, this name could be send across the wire and still the trading partner could uniquely identify this BIE. EbXML uses qualifiers to do this. A qualifier identifies the context of the BIE and must be unique for each BIE based on the same CC. Suppose we need a BIE to identify addresses in the

[1] This picture is a corrected version from figure 4-2 in [ccts]

Netherlands. A CC address already exists, and some geographical context has to be added. The qualifier that can be used here is NL, so the name of the BIE will be NL_ Address.

Context parameters like geographical location are hierarchical ordered. This ordering is necessary to determine whether or not a context is a subset of another context, and is usually described in a ISO standard, or some other public available list. The format of these hierarchies is not specified in a uniform way, and therefore these lists cannot be used. In order to come to an automated matching, this hierarchy has to be embedded in the qualifiers. In case of an address in the Netherlands, this means the BIE should be named: NL_ EUR_ Address. If this BIE is compared to a EUR_ Address, the subset relation can be derived from the qualifiers. This embedding of qualifiers is currently not part of the specifications, but should be added in order to come to an automated matching of documents.

5.2.4 Document assembly

In order to match activities, documents have to be matched. Figure 5.2 shows that documents are assembled from multiple ABIE's (Aggregated Business Information Entities) with some additional assembly information. The specification does not describe how this assembly takes place and what this extra information is, but refers to an old, outdated document. Core Components can never be used in document assembly. Currently, discussion is going on within the Core Component working group (part of the TMG working group of the UN/CEFACT) on how to assemble documents from BIE's. After consulting with a member of this group, the assumption is made that at top level, a document is an ABIE.

5.3 Using ebXML in the NUON - Manpower pilot

This section will treat the use of ebXML between NUON and Manpower, one of the pilots projects within openXchange.

NUON is the largest utility supplier in the Netherlands and uses a lot of temporary workers. Manpower, as a job agency, is supplier of temporary workers. NUON is large customer of Manpower. A large number of documents is exchanged between NUON and Manpower every week, including timecards and invoices. The pilot was intended to create a business case in which could be shown that automated exchange of these documents could save both companies a lot of money. To give an impression: NUON has a complete department correcting invoices send by Manpower. To reduce complexity of the pilot, the business case was reduced to exchange of timecards.

If the pilot points out that automated exchange of the timecards reduces faults, both NUON and Manpower plan to start using electronic document exchange with other companies as well. For this reason ebXML was chosen. The use of Core Components and Business Information Entities creates flexibility towards the future.

The first step in the project was to create a class diagram of the collaboration domain. In this project the Resource – Event – Agent or REA ontology is used [Car82], but other diagram techniques are allowed as well. The resulting diagram is shown in appendix D. Using REA diagram, Core Components and Business Information Entities are discovered. Figure 5.3 shows the resulting Core Components (names are simplified to improve reading). In this particular example, all Business Information Entities have the same properties as their corresponding Core Components. This is because of the fact that the Core Components were newly created according to the specific needs of NUON and Manpower.

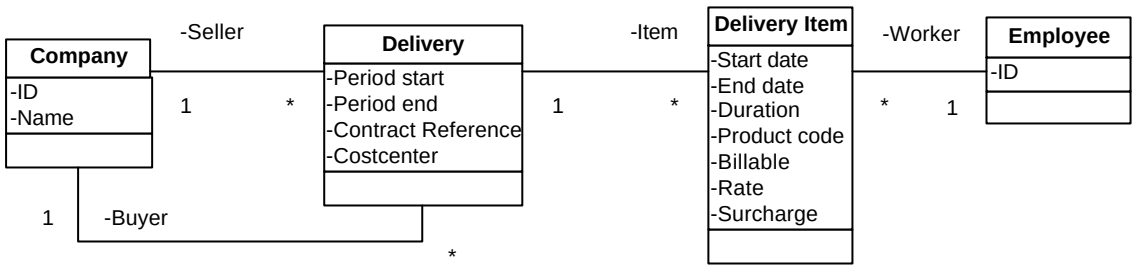


Figure 5.3: Pilot Core Components

In the context of temporary staffing, the timecard message contains a delivery is represented at top-level. A delivery is always between two companies, a buyer and a seller. In the context of temporary staffing, the hiring company (NUON) is the buyer and the job agency (Manpower) is the seller. A delivery consists of one or more items. In the context of temporary staffing, each delivery item is a period worked by an employee. Appendix E shows an example Timecard message

If another company wants to create its own Business Information Entity based on an existing Core Component, it may want to add (and remove) properties to the existing Core Component. A supplier of nuts and bolts may e.g. want to add the property "weight", while the property "rate" is of no value to him. Adding properties to a Core Components does not affect the Business Information Entities that are already based on the Core Component.

This reusability is one of the key features of ebXML. Within ebXML, a harmonization workgroup is created to achieve this reusability of Core Components and Business Information Entities. Both Core Components and Business Information Entities can be submitted to this workgroup using a special worksheet. Appendix C shows the worksheet containing the Core Components and Business Information Entities that was submitted to this workgroup as a result of the pilot. Eventually, all Core Components and Business Information Entities are published in a public registry.

5.4 Matching of documents

Since the assumption is made that a document is an ABIE at top level, the matching of documents is equal to the matching of two ABIE's. This matching problem is twofold. The syntax and cardinality of the ABIE's have to be matched, as well as the context the ABIE are valid in. In paragraph 5.3.2, the relation between a CC and a BIE was defined as 'a BIE is based on a CC'. In order to come to a meaningful matching, this "based on" will be described in more detail first. After this, the matching of context will be treated, followed by the matching of syntax. If both documents are exactly the same (same ABIE at top level), this document can be referenced to from the new formed bpss. If both documents differ, but the document of the sending party fulfils the information need of the receiving party, the document of the sender is referenced from the BPSS. To determine whether or not the information need is fulfilled, documents have to be matched.

5.4.1 Relation between CC and BIE

The relation between a CC and a BIE is said to be a "based on" relation. This relation is defined quite strict in the specification. For an ABIE, the relations means that an ABIE can only have properties that are based on properties of the ACC. This means an ABIE has the same or less number of properties as

the ACC. For a BBIE, as mentioned before, the relation means that its data type (or set of possible values) is smaller than that of the BCC it is based on. The based on relation for an ASBIE means that the ABIE it refers must be based on an ACC the ASCC refers.

The relation between an ABIE and an ACC has some resemblance with the inheritance construction in the UML, but there are some significant differences. Suppose we want to model three different types of products: medical, chemical and entertainment products. All products have in common that they have a product code and a description. The chemical and the medical products have a usage description while the entertainment products have an age indicator. Finally, the chemical products have chemical composition. All products have to be one of these three types. There can not be a direct instantiation of "product". This simple scenario can be modelled using inheritance in a UML class diagram as shown in figure 5.3 (a). The class "product" is an abstract class.

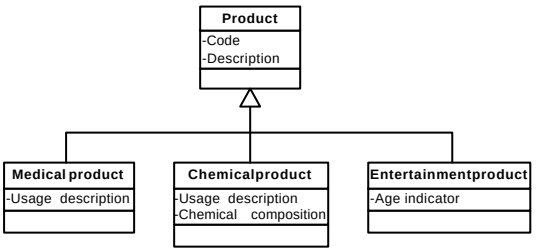


Figure 5.3(a): UML style

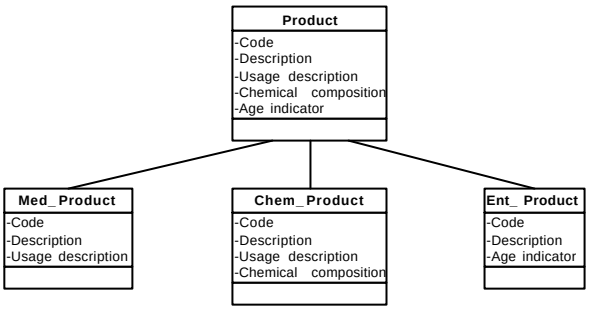


Figure 5.3(b): ebXML style

Figure 5.3(b) shows the way this is modelled in ebXML using a Core Component and Business Information Entities. The main difference between the ebXML diagram and the UML diagram is that the ACC has all the properties that exist in the ABIE's. Two reasons are often heard in favour of using this approach, instead of the UML way. First of all, in the UML diagram, there is no link between the two occurrences of "usage description". Therefor you can never be certain that these two properties are the same. A second reason is that, if there are a large number of ABIE's inheriting from an ACC, it will often happen that none of the properties exist in all ABIE's so in standard UML, the superclass would have no properties. The mayor drawback of this construction is that a specification has to be made that defines the inheritance relation (since it differs from the UML inheritance structure). Another drawback is that parameters in a ACC will tend to overlap each other if two ABIE's have properties that are similar but not equal.

If an ABIE "inherits" a property from an ACC, the data type may be changed (or "overwritten" in UML terms). In UML the data type can be changed in anything using overwriting. The relation between BCC's and BBIE's is must more stringent. The data type of a BBIE is always a restriction on the data type of the BCC it is based on, never an extension. If a BCC has a data type integer (1..100), a BIE based on this BCC may either have the same data type or a more restricted one e.g. integer (10..80). A data type integer (1..200) is not valid for this particular BCC.

5.4.2 Context matching

At top level, a document is an ABIE and each BIE exists, or is valid, in a specific context. This context must be described using context categories and gives meta data about the BIE. When a company is going to describe the documents used in its business process, the context categories can be used to check whether or not the necessary BIE's already exists. The concept of context is also used in the matching

process. When matching documents (or ABIE's) there must be some relation in context in order to come to a successful match.

But how is this context going to help in the matching of documents? Suppose somewhere in a business process, a company specifies to receive some document. This means that this company is receiving an ABIE that has a specific context, say A. The company that sends the documents also has a context specified on that particular ABIE, say B. The question is: when do these two contexts match? If both contexts are equal, A=B, than we could certainly speak of a match since all instantiations of these ABIE's are valid in both contexts. But what if the contexts are different? If both contexts are completely different, the matching should fail since there is no possible instantiation of this ABIE that is valid in both contexts. In case of an overlap in context, there are instantiation of the ABIE that are valid in both contexts, but not all instantiations are.

To solve the question on when different contexts can match the problem must be viewed from receiving party. This party specifies that it needs a particular piece of data in order to continue its business process. When two documents match, every document instantiation of the sending party must be valid in the context of the receiving party. This means the context of the document from the sending party must be part of the context of the receiving party. If context of a particular ABIE is considered as the definition of the set of all possible instantiations of that ABIE, the set of the sending party must be a subset of the set of the receiving party (see figure 5.4)

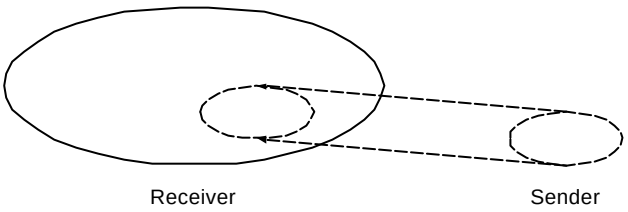


Figure 5.4: Context matching

Only when this condition is valid, a correct match should be given. The agreement file (or ABIE) should have the same context as the ABIE that is being send.

5.4.3 Matching syntax

Besides the context of all Business Information Entities, also the syntax has to be matched. The syntax of an ABIE defines the cardinality of its properties. All mandatory properties of an ABIE should at least be present (mandatory or optional) at the other ABIE. If a property is mandatory in one of the ABIE's, it should be mandatory in the agreement document. If a property is optional in one of both and mandatory in the other, it must also be mandatory in the agreement ABIE. All possible combinations are described in table 5.1.

		Sender		
		M	**O**	**A**
Receiv	**M**	M	M	N!
	O	M	O	A
	A	N!	A	-

M: Mandatory
O: Optional
A: Absent
N! No Match!
-: Will not occur

Table 5.1: Truth table context matching

On BBIE level, the data types of both BBIE's need to be matched. This matching is similar to that of context matching. If the set of possible values of the sender is a subset of the set of possible values of the

receiver, the match should resolve to true. If not, human involvement is required and agreements have to be made on if and how the data type should be translated.

5.5 Matching of conditions

EbXML offers the possibility to specify conditions on activities. These conditions include pre and post conditions, "begins when" and "ends when". Although matching conditions is essential for fully automated matching, it is omitted from this assignment for the following reasons:

- The specifications are unclear about the use of these conditions. Sometimes they are used for effect definition (if the pre condition is valid when the activity starts, then the post conditions will also be valid), and sometimes they are used to influence the choreography (if a precondition is not valid, the activity will not start).

- If pre and post conditions are to be matched, they have to be interpreted by a computer. Although ebXML recommends the use of OCL, users may chose any language to specify their conditions. This makes it nearly impossible to match these conditions in an automated way.

- Even if the intended use is clear and OCL is required, the matching of conditions is extremely difficult. The most obvious way for matching is combining expressions, e.g. (x<5) combined with (y> 7) results in (x<5 AND y>7). If conditions are about the same variable, the expression has to be interpreted and rewritten, e.g. (x>3) combined with (x<7) results in (7<x<3). The matching algorithm should detect if the resulting expression can never evaluate to true, e.g. (x<3 AND x>7). If two different expressions restrict the same variable, this can still be done but if they restrict different parameters that are somewhat related, it is nearly impossible to compare these in an automated way.

For the time being, conditions will be ignored in the matching process. Until the mentioned problems are solved, human involvement will be required when matching conditions.

6 CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Since the late 80's, EDI has proven the value of electronic business. Since EDI is not affordable by a majority of the companies, a lot of research has been done on new electronic business standards that rely on XML and the Internet. EbXML distinguishes itself from other standards like HrXML and UBL by providing an entire framework for business to business ecommerce, rather than just a messaging standard.

Besides offering more functionality than other standards, ebXML tries to offer a standard that is very flexible towards its users. The downside to this increase in functionality and flexibility is an increase in complexity. Not only complex in the way that there are about 25 specifications to describe the entire standard, but also the content of some of the specifications is very complex. Besides complexity there are still a significant number of errors, inconsistencies in the specifications. Moreover, sometimes the content of the a specification can be interpreted in multiple ways.

More and more companies start implementing and using ebXML, but because of reason mentioned earlier, only a part of the framework is used. Most companies only use the messaging service. A few companies use the BPSS, but the Core Components are almost never used. This of course is too bad, since both BPSS and Core Components have an added value compared to other e-business standards.

Within the ebXML working groups that create the specifications, there are roughly two visions on how to do business in an electronic way. One vision, often shared by people that have an extensive EDI background, is document oriented. In this vision, two companies are strictly separated and documents are the fundaments of doing business. In ebXML, documents are no more than electronic variants of paper business documents like order, invoice, etc. The other vision is much more object oriented. In this vision, the collaboration between two companies can be seen as an object model in which both companies participate. Both companies have a view on the collaboration and documents are used to synchronize the view both companies have on the collaboration. Both visions have to be brought together in order to create clear and correct specifications.

In this research, a begin has been made on automated matching of business processes. In order to use the BPSS on a large scale, automated matching is necessary. Although the first results are promising, additional research is necessary to come to a system that can match business processes in a fully automated way. Especially the matching of pre and post conditions is crucial for process matching.

6.2 Recommendations for future work

In order for ebXML to become the new standard for doing electronic business, the specification must be accessible for a larger group of people. Currently, a lot of work is done in order to remove the errors and inconsistencies. Once this is done, efforts should be made to make the specification easier to read.

The BPSS specification has a unclear semantics on certain concepts and therefore it can sometimes be interpreted in several ways. In ebXML transition to another activity is sometimes forced while that activity is not finished yet. According to the UML activity diagram specifications, this is not allowed. Another issue is the use of pre and post conditions. The specification is unclear about the use of conditions and this should be clarified.

The Core Components have reached the point in which the first set of standardised Core Components and Business Information Entities are to be created from submitted proposals. For the purpose of matching, it is very important to detect a hierarchy in context values. Qualifiers should be used to specify this hierarchy, otherwise matching of Business Information Entities will become unnecessary complex.

Further research on the matching of business processes is needed. In particular the matching of pre and post conditions is crucial to come to a successful matching algorithm, but this is also a difficult task. The BPSS team should force the use of one specific language to specify conditions (preferably OCL) because comparing conditions of different languages is nearly impossible. Besides matching of conditions, also parameters on activities (like

Currently research is done on the matching of two CPP's for automated generation of a CPA. The matching of business processes should become a part of the matching of CPP's since both CPP and CPA reference business processes.

REFERENCES

[Blo92] Blommestein, F.B.E. van, *Barcodes & EDI*, Stenfert kroese uitgevers, 1992, 90-207-2251-4

[Bps02] EbXML business process specification schema version 1.05, July 2002

[Car82] McCarthy, W.E. The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. In *The Accounting Review*, pages 554-78, July 1982

[Cct02] EbXML Core Components Technical Specification version 1.90, December 2002

[Ebx02] www.ebxml.org, ebXML project homepage, February 2003

[Esh02] Eshuis, R, *Semantics and verification of UML activity diagrams for workflow modelling*, 2002, 90-365-1820-2

[Hrxml] www.hr-xml.org, homepage of HR-XML consortium, August 2003

[Pra91] Pratt, V.R, Modeling concurrency with geometry. In *Proc. 18th Ann. ACM Symposium on Principles of Programming Languages*, pages 311-322, January 1991.

[Ram99] Raman, D, *XML/EDI: Cyber Assisted Business in Practice*, 1999, 90-805233-2-1

[Sud97] Sudkamp, T.A, *Languages and Machines: an introduction to the theory of computer science - second edition*, Addison Wesley Longman Inc, 1997, 0-201-82136-2

[Ubl03] http://www.oasis-open.org/committees/ubl, homepage of UBL Technical Committee, August 2003

[Whi00] Enabling electronic business with ebXML, December 2002, available at: http://www.ebxml.org/white_papers/whitepaper.htm

APPENDIX A PROJECT DETAILS

This research is done as a thesis assignment part of the programme “technische informatica” (technical computer science) at the University of Twente. It will be conducted at TNO (Netherlands Organization for Applied Scientific Research), at the group E-Business in Enschede.

Description of TNO E-business

This group is part of the TNO Physics and Electronics Laboratory institute (TNO-FEL), division Telecommunication and Security. TNO provides a link between fundamental research as a source of knowledge and practical application as the use of this knowledge in real life situations like commercial activities. The core activities of TNO are:

- development of knowledge;
- utilisation of knowledge for clients in industry and government;
- technology transfer, especially to small and medium-sized enterprises (SME's);

TNO-FEL is the largest independent ICT-lab of the Netherlands. TNO E-Business is a relatively young part of this institute and is situated in Enschede, near the University of Twente and BSC (Business and Science Park). The products of TNO E-Business can be divided in three groups: development, analysis, and knowledge transfer. Under the authority of and in co-operation with companies, governments, and other knowledge bodies, TNO E-Business develops:

- architectures and concepts for innovative e-business applications on the levels of business processes, information, components and systems;
- architectures for ICT-support of innovative e-business models, covering the same levels - new technologies for e-business;
- demonstrators and prototypes of innovative e-business applications;
- applications standards for e-business, whether or not within certain fields or application areas.

Furthermore TNO E-Business offers:

- requirements analysis for e-business applications;
- impact analysis of e-business applications and technologies;
- counterchecks, second opinions and verification of e-business applications, specifications and architectures.

Besides, TNO E-Business occasionally offers knowledge transfer on a customized basis, by means of seminars, courses and the like.

Project supervision

Supervision will be done by ir. Erwin Folmer of TNO E-Business and ir. Fred van Blommestein of Berenschot. The supervision of the University of Twente will be performed by the group Information Systems (IS) of the Faculty of Computer Science, in the persons of prof.dr. Roel Wieringa and dr. Pascal van Eck.

Research plan

In order to answer the research questions, and come to a working algorithm, the following activities have to be done:

- A literature search on activity diagrams and ebXML
- Interviewing experts on activity diagrams and ebXML
- Design the algorithm
- Implement the algorithm
- Test the algorithm

The following gantt chart will indicate when the activities will take place.

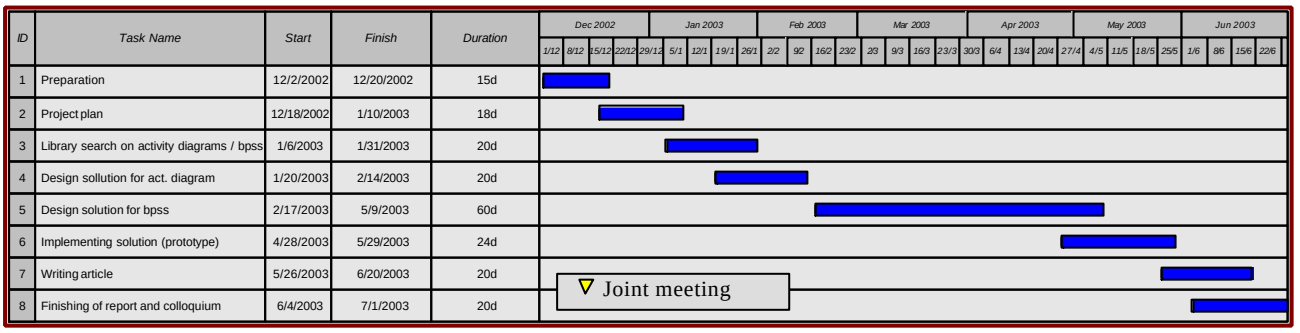


Figure A1: Time planning

Project deliverables

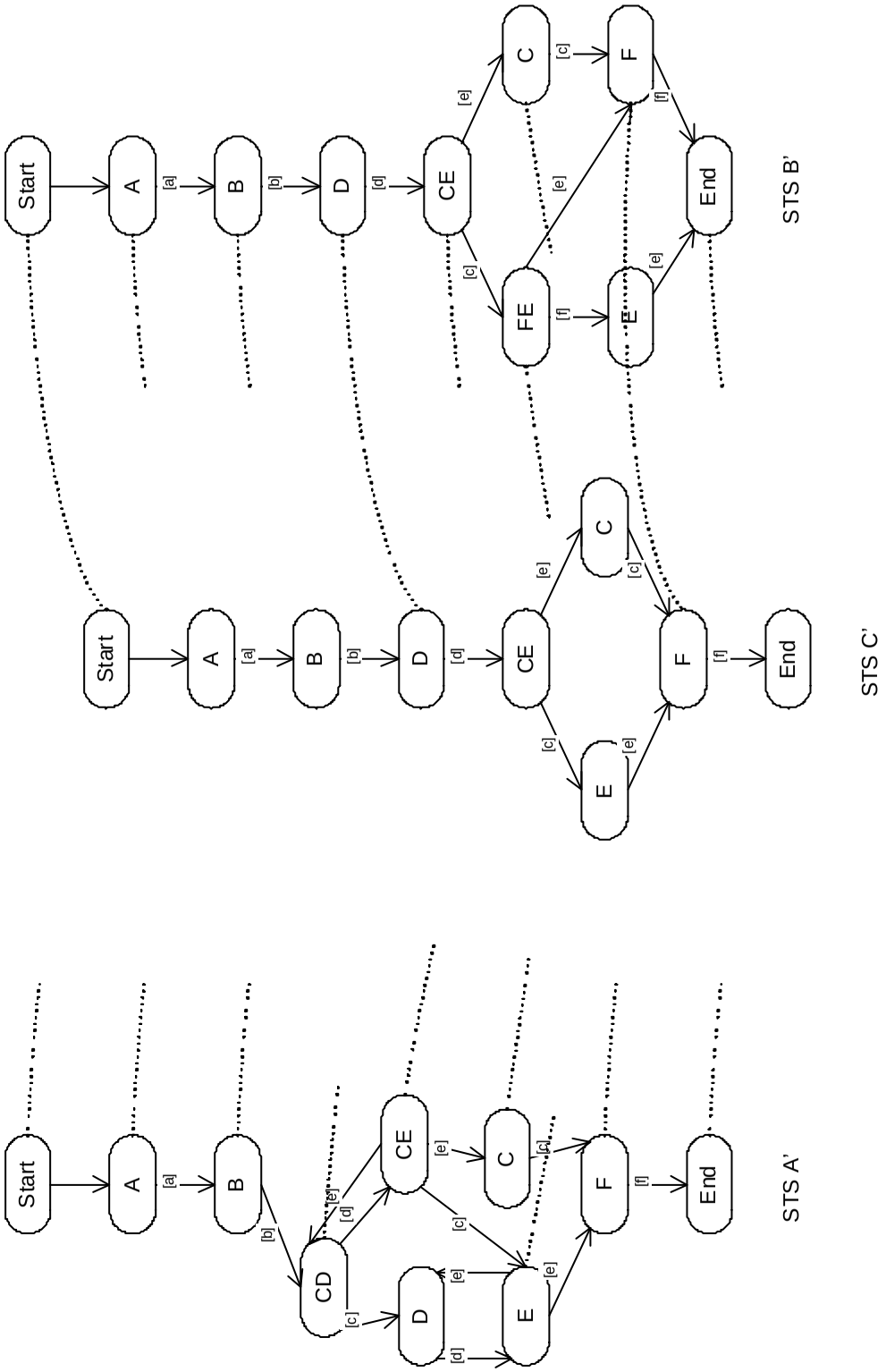
In addition to the thesis assignment rapport and the colloquium, some other deliverables can be identified. All of the deliverables are shown in following table:

Project deliverable	Description
Report	Thesis assignment report
Colloquium	Academic lecture and discussion
Prototype	An working implementation of the algorithm
Article	Writing about encountered problems en found solution

Table A1: Project deliverables

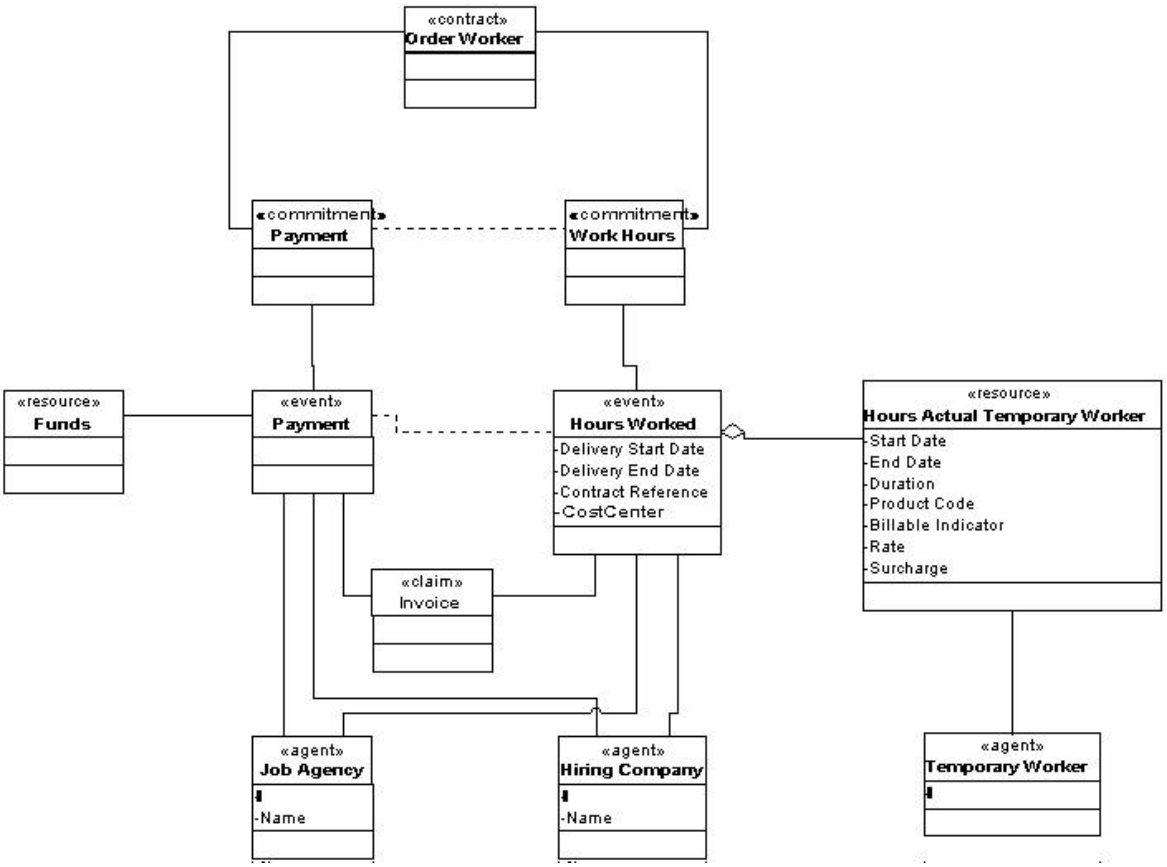
APPENDIX B EXAMPLE OF SIMULATION

The following pictures shows a graphical representation of the simulation relation between the graphs in figure 4.9 and figure 4.10



APPENDIX C HARMONISATION WORKSHEET

APPENDIX D NUON – MANPOWER REA MODEL



APPENDIX E EXAMPLE TIMECARD MESSAGE

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com) by Dennis Krukkert (TNO Fysisch en Elektronisch Laboratorium) -->
<!--Sample XML file generated by XMLSPY v5 rel. 3 U (http://www.xmlspy.com)-->
<Timesheet xmlns:ccts="urn:oasis:names:tc:ubl:CoreComponentParameters:1.0:0.70" xmlns:oxcc="urn:openXchange:CoreComponents:1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="TimeSheet.xsd">
        <TemporaryStaffing_Delivery.PeriodStart.DateTime>2003-03-01T08:00:00</TemporaryStaffing_Delivery.PeriodStart.DateTime>
        <TemporaryStaffing_Delivery.PeriodEnd.DateTime>2003-03-01T17:00:00</TemporaryStaffing_Delivery.PeriodEnd.DateTime>
        <TemporaryStaffing_Delivery.ContractReference.Identifier>85658945</TemporaryStaffing_Delivery.ContractReference.Identifier>
        <oxcc:TemporaryStaffingActual_DeliveryItem.Details>
                <TemporaryStaffingActual_DeliveryItem.StartDateTime.DateTime>2003-03-
01T08:00:00</TemporaryStaffingActual_DeliveryItem.StartDateTime.DateTime>
                <TemporaryStaffingActual_DeliveryItem.EndDateTime.DateTime>2003-03-
01T17:00:00</TemporaryStaffingActual_DeliveryItem.EndDateTime.DateTime>
                <TemporaryStaffingActual_DeliveryItem.Duration.Measure
unitCode="Hours">8</TemporaryStaffingActual_DeliveryItem.Duration.Measure>
                <TemporaryStaffingActual_DeliveryItem.ProductCode.Code>Regular</TemporaryStaffingActual_DeliveryItem.ProductCode.Code>
                <TemporaryStaffingActual_DeliveryItem.Billable.Indicator>true</TemporaryStaffingActual_DeliveryItem.Billable.Indicator>
                <TemporaryStaffingActual_DeliveryItem.Rate.Amount currencyID="EUR">30</TemporaryStaffingActual_DeliveryItem.Rate.Amount>
                <oxcc:Employee.Details>
                        <Employee.Id.Numeric>100</Employee.Id.Numeric>
                </oxcc:Employee.Details>
        </oxcc:TemporaryStaffingActual_DeliveryItem.Details>
        <oxcc:TemporaryStaffingActual_DeliveryItem.Details>
                <TemporaryStaffingActual_DeliveryItem.StartDateTime.DateTime>2003-03-
15T08:00:00</TemporaryStaffingActual_DeliveryItem.StartDateTime.DateTime>
                <TemporaryStaffingActual_DeliveryItem.EndDateTime.DateTime>2003-03-
15T17:00:00</TemporaryStaffingActual_DeliveryItem.EndDateTime.DateTime>
                <TemporaryStaffingActual_DeliveryItem.Duration.Measure
unitCode="hours">8</TemporaryStaffingActual_DeliveryItem.Duration.Measure>
                <TemporaryStaffingActual_DeliveryItem.ProductCode.Code>Sickness</TemporaryStaffingActual_DeliveryItem.ProductCode.Code>
                <TemporaryStaffingActual_DeliveryItem.Billable.Indicator>true</TemporaryStaffingActual_DeliveryItem.Billable.Indicator>
                <TemporaryStaffingActual_DeliveryItem.Rate.Amount currencyID="EUR">30</TemporaryStaffingActual_DeliveryItem.Rate.Amount>
                <oxcc:Employee.Details>
                        <Employee.Id.Numeric>100</Employee.Id.Numeric>
                </oxcc:Employee.Details>
        </oxcc:TemporaryStaffingActual_DeliveryItem.Details>
        <oxcc:Buyer_Company.Details>
                <Buyer_Company.Id.Numeric>100</Buyer_Company.Id.Numeric>
                <Buyer_Company.Name.Text>Nuon</Buyer_Company.Name.Text>
        </oxcc:Buyer_Company.Details>
        <oxcc:Seller_Company.Details>
                <Seller_Company.Id.Numeric>200</Seller_Company.Id.Numeric>
                <Seller_Company.Name.Text>Manpower</Seller_Company.Name.Text>
        </oxcc:Seller_Company.Details>
```