ELP

towards an extendible logistics protocol

Snoek, M.L.

Award date:
2009

ELP

Towards an

Extendible Logistics Protocol

April 2009

Eindhoven University of Technology
Department of Mathematics and Computer Science
Supervisor: dr. A.T.M. Aerts

Thesis author: M.L. Snoek (0474935)

Global Data Exchange

TU/e Technische Universiteit Eindhoven University of Technology

Acknowledgment

I would like to express my profound gratefulness to my supervisor, dr. Ad Aerts, of the Mathematics and Computer Science department of the Eindhoven University of Technology. His guidance, extensive support, recommendations, patience, comments and proofreading throughout my graduation and writing of this thesis are invaluable. I will especially remember his continuous support despite of the long period it took to complete this thesis. This support even extended to Switzerland during his research at CERN. My visit to CERN and Geneva has been a valuable experience that I will not forget.

I would like to thank Global Data Exchange in Maarssen for the opportunity to make my graduation possible on the subject of the ELP. My sincere gratitude goes out to drs. Laurens van Run for his recommendations and detailed proofreading of this thesis.

I would like to express my gratitude to my parents, Iede and Henriette, for making my study at the Eindhoven University of Technology possible and especially for their support during my graduation. I would like to thank my sweet girlfriend Jessica for her endless encouragement and patience during the writing of this thesis.

Mark Snoek

Table of contents

List of illustrations

1 Abstract

The transportation industry worldwide consists of many transport companies using information systems, namely Transport Management Systems (TMS), to increase their efficiency, service and profit margin. Transport companies cooperate by outsourcing orders to realize cost savings, additional specialisms and flexible capacity. Unfortunately, the advantages of TMS software products are limited to situations without outsourcing and can be increased when software products support a standard to exchange information. This thesis covers the research of how existing TMS software products can be extended to support the exchange of information to maximize the advantages of outsourcing and the use of information systems. This research focuses primarily on courier companies which are a part the transportation industry. The approach of the research consists of analyzing existing solutions, business processes and information exchanged during these business processes. Using the results, an information system, namely ELP, has been designed. This design includes technical details to exchange information using distributed database technology, including limited support to extend it with proprietary elements. The design of ELP has been used to develop a prototype to test key functionality of the design. This prototype and the design of ELP show that the approach provides most of the required information to create an extension for existing software products to exchange information when orders are outsourced. Finally, additional topics are provided that need to be researched before one being able to create a full implementation of the design and to solve several practical issues which remained unsolved.

2 Introduction

2.1 Research area and background information

Transportation of goods one way or another effects everybody. Without it, stocks would be empty and a lot of people would be unemployed. The research area of this document lies within this important industry. The transportation industry exists of many kinds of transports, such as transport over water, rail, roads and through the air. The focus of this reports is limited to transport over the road with specific attention to courier companies.

This chapter contains an introduction to transportation in general as well as an introduction to IT solutions that are used to provide automation of transport companies.

2.1.1 Transportation in general

Typical transportation process

Every day many parcels, pallets, boxes, etc are picked-up and delivered by thousands of courier and transport companies worldwide. The common characteristic of these companies is that they pick-up some good at one location and deliver it at another. The goods can be transported directly from the pick-up location to the delivery location, but it also happens frequently that the goods are stored at intermediate storage facilities, which are known as warehouses. Goods can be stored in these facilities for a short or longer period. If intermediate storage facilities are used then the goods will be transported from the pick-up location to a warehouse, optionally to another warehouse or warehouses, and finally to the delivery location. In short, it can be said that a good is picked-up at a first location, transported to zero or more warehouses and finally delivered at the delivery location. Typical courier companies that use a lot of warehouses worldwide are UPS, DHL, FedEx and TNT. The transportation of a parcel from Amsterdam to Sydney will probably not be executed without intermediate storage in a warehouse at, typically, an airport.

Types of transport companies

Within the large group of transport companies there are several types that can be distinguished by their kind of clients. There are so called “charter” companies, with usually a few cars or trucks, that only execute orders for other transport companies. Their transportation means usually have no company name printed on it -they use so-called “white-label” vehicles- to deliberately be not tied to a specific company. The second type of company also has regular customers, which means that its customers are other transport companies as well as regular customers. A third type of transport company has mostly regular customers. Yet another type of company, which in fact is not a transport company, is the transport broker. Brokers accept orders from transport companies or regular clients that are outsourced to a (specialized) transport company. Brokers usually have a large network of transport companies that can perform the actual transport.

Outsourcing is generally accepted

From the variety of transport companies, especially the charters, it can be concluded that the execution of an order is frequently outsourced, otherwise these companies wouldn't exist. In fact, the transport brokers can be seen as transport companies that only outsource their orders. One of the reasons for companies to outsource an order is that they are not specialized enough to execute the order itself. Another, more important, reason is cost saving. Although outsourcing is usually more expensive, because of an extra company that would like to make a profit, it can be a cost saver for transport companies as is illustrated in the following examples.

Charters are mostly used when the capacity of a transport company itself has been reached. It is more expensive to have an extra vehicle, that is only used during busy periods, in a fleet of vehicles than to pay the extra cost of a charter. The charter can be seen as shared capacity

among transport companies with a slightly higher price per kilometer.

Another example is the extra cost due to driving with an empty vehicle. When a vehicle has delivered its goods it is available for a new load. It can be used by other transport companies that have to pick-up goods in the destination area of the empty vehicle and so avoid sending one of their own vehicles. In this way the owner of the vehicle makes an extra profit and the outsourcing company has fewer costs. There are specialized IT companies that provide information about the locations and vehicles between transport companies such as [Intellicom] and [CourierExchange].

Due to the large competition, the rates of transport companies are under pressure. Outsourcing is a possibility to save costs and therefore possibly make a profit. It also provides a higher 'virtual' capacity with optionally more specialization.

Transportation process and information sharing

Courier companies frequently offer Track&Trace solutions to their customers to keep track of the transportation of their goods. They, for example, provide information about the current and past locations of a parcel, the direction it is going and finally the person who signed for the delivery. These systems are needed for Just In Time (JIT) delivery and possible high priority express delivery services. If a courier company performs the whole transportation process itself then this Track&Trace information is usually detailed, up-to-date and complete. Unfortunately, most courier companies don't always take care of the whole transportation process themselves. The entire transportation process or just a part of it can be outsourced to other (transport) companies. This outsourcing can have an effect on the accuracy and detail of the Track&Trace information when not all transport companies have compatible Track&Trace information systems to exchange information, if they have any at all.

In short, it can be said that the lack of information systems that can successfully exchange transportation details is an impediment for the interoperability and the availability of up-to-date shipping data that is of interest to more than one transport company.

Combined shipments

During the transportation from the original location to the destination, multiple goods can be combined using special resources for a part or the total of the transportation track. Typical examples of these resources are pallets and sea containers. The resources are goods themselves from a transport company point of view. If a transportation resource is moved from its original location, this is the location where it is loaded, to the final destination, this is the location where it is unloaded, then all contained goods have the same location properties as the transportation resource. Pallets and containers don't necessarily need to be resources. This depends on the fact if a container is a pure wrapper that is also delivered or a wrapper that is introduced by the transport company to accommodate goods during transport.

2.1.2 Transport from an Information Technology point of view

Automation of transport companies is done by many IT companies around the world. Since there are many transport companies, the market for IT companies supplying software is also extensive. A survey of TLN (“Transport en Logistiek Nederland”) [TLNNOV06], which is the branch organization of transport and logistic companies in The Netherlands, indicates that in 2006 there were 36 automation companies with a specialized product for the transportation sector only in the Netherlands. Transport companies can be found all over the world and one can conclude that there are hundreds of software products available for this sector today. The Netherlands plays an important role in the European market of transportation needs due to the port of Rotterdam and Schiphol airport.

Transport companies around the world are storing information about their business in their

information systems. As mentioned, these systems are build by a lot of different manufacturers and all have their own specific ways of storing information. When transport companies would like to improve their collaboration then they should be able to exchange information about the goods they are transporting. When two information systems don't understand each others information then this can lead to difficult or even dangerous situations.

The automation companies are competing with specialized software for specialized kinds of transport. Examples of these specialization are trailer transport, courier/express, distribution, dangerous good transport and railway transport. All these specializations have aspects in common such as loading, unloading and actual transport of goods. Since one of the practical aspects of transportation is outsourcing, it would be preferable when the systems of all these suppliers can communicate using a common language. Communication between these software solutions is called Electronic Data Interchange, also known as EDI.

A look at the website of [TLNNOV06] companies shows that a number of companies has integrated EDI functionality into their software product. Unfortunately this functionality seems to be limited to the import and/or export of orders. Although this is very useful functionality in the sense of preventing errors and avoiding repetitive input work, the approach is very basic. It doesn't provide real-time or near real-time exchange of information nor is the exchange automated. Another limitation seems to be the proprietary nature of the formats that are used to exchange the information. This can be concluded from the fact that no supplier presents any schemes needed to format data for import or export. The inability to exchange data between transport companies and clients in a universal way was a motivation to start doing research on this subject at Global Data Exchange B.V., located in Maarssen, The Netherlands.

Global Data Exchange originated from two companies that merged in 2004. These two companies both produced an information system for courier companies. The merger was a good moment to examine the possible data exchange between the two existing software products that are targeted for the same market. Since the two information systems both stored data of courier companies one might think that the data structures of the databases behind the interfaces were alike. In fact, the databases did have a lot aspects in common. For example, both software programs used data structures for clients, orders, invoices and rates. They also had comparable relations between entities such as one-to-one and one-to many. The biggest difference in the data structures was the level of detail of stored items and the possible flexibility that the software would like to offer to the user. An example of the level of detail is the extension of a house number, such as 'BIS' or 'II', that is a separate field in one program, but is assumed to be included in the house number by the other software package. An example of the flexibility is the aspect of picking up multiple boxes and deliver them at different addresses (distribution). One software package assumed these are multiple orders while the other can handle it as one order.

When even two software products, that are targeted for the same market, have a collection of aspects that are not common then it would be interesting to know where deviation takes place and how the design of these product can be used to create a more common framework to exchange data. It seems a challenge to design a framework that can bring Transport Management Software (TMS) products closer to each other from a technological point of view and to create a solution that makes them compatible.

2.2 Reason for research

Many courier and transport companies exist worldwide. Some of them have enough resources to provide a worldwide delivery coverage, but most don't. Also, some of them have enough capacity and specialization to execute all transports themselves, but most don't. These conclusions can be drawn from the knowledge available within Global Data Exchange about the courier and transport market. Many courier and transport companies in The Netherlands work together to execute

transports that they cannot execute on their own or to make a better profit. When these benefits are available in a country as The Netherlands then it can be assumed that the benefits of outsourcing are also available in other countries. The reasons for this are the geological aspects and prosperity of other countries that are comparable to The Netherlands. Transportation in general can be seen a way to bring goods from a supplier (ports, airports, factories) to a demander (people, companies, factories). A high population implies a high demand and thus a lot of transportation where a high prosperity suggests the relative luxury of couriers transporting goods. Countries that can obviously be compared to The Netherlands are those in West Europe and several other parts of the world, such as the United States of America. The benefits of outsourcing in these countries can even be bigger, because a larger country size suggests transports over a longer distance.

With so many courier and transport companies around and so many of them working together, one could ask whether their information systems also support this collaboration. From the information available at Global Data Exchange which is a specialized software supplier for the courier industry and market leader in The Netherlands, this question can be answered negatively. Several questions that arise are:

- Why don't these information systems support some kind of universal outsourcing functionality?
- Are there shortcomings in the current standards that prevent this?
- Do standards even exist?
- What aspects should be included in this universal outsourcing functionality?

The summarizing term “transportation” is a frequently used term that is so wide that restrictions are needed to answer the questions above in a reasonable amount of time. For this reason, the domain of these questions and the research question, given in the next chapter, is restricted to the transportation that is done by courier companies. Consequently, the domain does not include:

- Transportation of raw materials
- Transportation of humans or animals
- Transportation that is not executed by vehicles

Additionally, the domain is restricted to the physical aspect of transporting goods. For example, the domain does include packed goods, vehicles and companies that own the vehicles, but does not include government issues, customs and financial aspects.

Although the domain is now limited to courier companies, this doesn't explicitly exclude from being applicable for other kinds of transport companies. When these companies only transport packed goods then they would probably also fit within the domain as it doesn't prescribe anything about sizes or weight. Instead of referring to courier companies, the following chapters refer to transport companies because a courier company is in fact a specialized transport company and other transport companies are not excluded from the domain explicitly.

3 Research question

3.1 Outsourcing support for information systems

From the introduction it can be concluded that the two software products of Global Data Exchange do not have a common way of exchanging information between them. Other software products don't seem to have functionality implemented to automatically exchange information between them. Since all of these products and their targeted markets will require specific functionality for information exchange the following research question is raised:

How can an information system be designed that provides general functionality to exchange information about the execution of the transport of goods and give the possibility to extend it with proprietary elements?

This research question is quite general and cannot easily be answered without asking several subquestions.

To exchange information there has to a language, consisting of a syntax, semantics and synchronization rules, that communicating participants all understand and support. Since a communication language between information systems is called a protocol, the subject of this document defined as the Extendible Logistics Protocol (ELP), where the term Extendible will be illustrated later.

3.2 Derived subquestions

Some subquestions can be answered immediately while others are answered throughout this document or remain unanswered. Unanswered or partially answered questions can remain due to difficulties to answer them, being out of scope, being less relevant than others or other reasons. Unanswered question can be seen as a base for future work.

One important subquestion that can be asked is: what is the scope of ELP?
The scope of ELP defines the context wherein ELP can be used and includes information about information about the intended industry, intended users, with relationships and interaction between them, as well as business processes, divided into categories, that they execute and are part of ELPs functionality.

The intended industry of the scope is already mentioned by the domain in the previous paragraph, namely the courier industry that is part of the transportation industry. The intended users are courier companies and participants that appear in their business processes. These participants can be, but are not limited to, clients, namely companies or natural persons, transport brokers as well as other courier companies. The previous paragraph already mentioned that courier companies are considered (specialized) transport companies. The following table presents the intended users, their roles and relationships.

Q201

User	Role	Relationships
Transport company	An entity that provides and executes the service to physically transports goods from one location to another using using road vehicles.	A transport company offers its services to clients, brokers and other transport companies. It accepts orders from clients and is held accountable by clients. It can outsource orders to transport companies or brokers. In this case its role becomes Client.
Client	An entity that would like some goods to be transported from one location to another.	A clients places orders at a transport company or broker.
Broker	An entity that provides at least the same services as a transport company, but doesn't physically transport goods.	A broker places orders at transport companies or other brokers. It accepts orders from clients and is held accountable by clients. It only outsources its orders to transport companies where its role becomes Client.

Table 3.1 – Users, roles and their relationships

The role of broker is in fact a combination of transport company and client and therefore not mentioned specifically anymore.

The domain in the previous paragraph is restricted to the physical aspect of transporting goods. This implies that the supported business processes within the scope only apply to this aspect and therefore include order placement and track and trace, but exclude financial or legal aspects.

The following subquestions are not answered immediately, but most of them will be answered throughout this document. Appendix A presents an overview of which questions are answered in which paragraph. If a paragraph focuses on a subject that is related to one or more of the following subquestions, a rectangle with the a question mark and the number(s) of the subquestion(s) is given on the right of that paragraph. The rectangle on the right of this paragraph is an example.

Q999

Subquestions related to operational matters

[Q001] Why would users like to exchange information?
[Q002] Which business processes are the users involved in?
[Q003] What information is going to be exchanged during the business processes?
[Q004] What responsibility during conducting business processes does every participant have and are these responsibilities equally distributed?
[Q005] How valuable is an information system to exchange information to the participants?
[Q006] How is the ownership of information organized?
[Q007] What legal aspects, such as confidentiality, authentication and digital signatures, are involved in the business processes?

Subquestions related to external and financial matters

[Q101] What solutions are currently available?
[Q102] Is there any need for a new information system?
[Q103] What is the maturity and acceptance of existing solutions?
[Q104] Which properties of existing solutions are desired in a new information system?
[Q105] Which desired properties of a new information system existing solutions not provide?
[Q106] How compatible should a new information system be with existing information systems?
[Q107] What barriers can be expected for a new information system to be accepted?
[Q108] Which investments are required for a new information system compared to existing solutions?

[Q109] What legal aspects, such as licenses and patents, are involved?

Subquestions related to limitations and extensions

[Q201] Is this system only applicable within the transportation industry?
[Q202] What extendability can be expected of ELP?
[Q203] Is is possible to design the system in such a way that it provides functionality to exchange business process information in general, for example by introducing multiple layers?
[Q204] How can the information system be designed to not strictly limit its participants to standard business processes to increase acceptance and compatibility?

Subquestions related to data quality assurance

[Q301] What are the requirements for availability, security, accuracy and performance of the exchange of information?
[Q302] Is it possible that participants do not agree on the information they exchange and how can these conflict be prevented or solved?
[Q303] How can a participant continue to work while not being able communicate with other participants and are there any limitations to this?
[Q304] How can it be prevented that all participants fully rely on the other participants being available?

Subquestions related to extending exchange of information (outsourcing)

[Q401] Are business processes limited to an exact number of participants?
[Q402] How can participants be added to business processes?
[Q403] How is the responsibility organized when a participant would like to add another participant that is unknown to the existing participants?
[Q404] How are the rights and relationships between participants managed?
[Q405] How is the administration of participants set-up?
[Q406a] Does every participant know about all other participants?
[Q406b] Is it required that every participant is able to communicate with all other participants for every business process?

Subquestions related to technological aspects

[Q501] How is communication between participants set-up?
[Q502] What kinds of communication means are suitable?
[Q503] What are the consequences if the information system fails?
[Q504] Which techniques can be used to exchange information between participants?
[Q505] What are the consequences of different locale settings worldwide?
[Q506] Is is possible to supply ELP functionality as middleware?
[Q507] Which existing technological standards can be used to simplify implementations and increase compatibility?
[Q508] Are centralized external coordinators needed or can they be avoided?

3.3 Summary

There appears to be no common available solution for TMS software to exchange information preserving proprietary elements, especially when it comes to outsourcing between transportation companies. This leads to the research question: how can an information system be designed that provides general functionality to exchange information about the execution of the transport of goods and give the possibility to extend it with proprietary elements? The name of this information system is defined as the Extendible Logistics Protocol, abbreviated as ELP.

4 The approach to the ELP design

To give an answer to the research question and the subquestions that are raised, the first step is to define the functional requirements of the information system. This chapter first focuses on the functional requirements of the system by the clients, who place orders, and transport companies, that execute the orders. These requirements are needed to be able to tell whether a, new or existing, information system provides the functionality or at least indicates which functional requirements are missing. After comparing the functional requirements to existing information systems, this chapter describes the steps that are taken to design an information system, ELP, with transport companies in mind. Although it is primarily designed for usage by these companies, it should be able to be altered to use specific techniques within information systems for other (industrial) areas.

4.1 Functional requirements

To be able to design an information system that can solve automation problems or introduce new functionality, it has to be known what functional requirements are required by the users of the system. The functional requirements can split into multiple categories, namely:

Business processes: functional requirements that belong to this category are based on day-to-day operations, such as getting quotations and placing orders.
Information management: functional requirements that involve storing and retrieving information from a (local) information system. This is information should support the business processes.
Management information: functional requirements for managers and board members to be able to retrieve management reports.
Legal issues: functional requirements that belong to this category describe support for legal operations, obligations and documents that are required for governments and customs.
Track & Trace: functional requirements that belong to this category describe functionality to be able to get an up-to-date view on the progress orders being executed. This category is added separately from the business processes, because it provides clear functionality of the exchange of information to keep it up-to-date as described in chapter 2.

4.1.1 Business processes

Q001
Q002

Requesting and providing quotations
The client would like to be able to receive a quotation of the transport company about the costs to transport the goods from the pick-up location to the final destination. This quotation is based on the the physical aspects of the goods as well as the preferred time windows and optional dangerous goods indications. When the transport company receives a quotation request from a client then it would like to be able to answer this request with a financial proposal. This quotation has a period of validity. If the transport company is not able to execute the order then the client would like to receive a rejection including reasons why the transport company wasn't able to create a quotation. Optionally, it is desired that the transport company can give a request with an alternative proposal, such as slightly changed time windows. [RQFuncBus1]

Get quotations from other transport companies
Apart from the client, the transport company would like to be able to outsource (parts of) the transport of a placed order. Before deciding whether to use outsourcing, the transport company would like to be able to request and receive quotations from other transport companies. The received quotations can be used to create a quotation to a client. When the company requests quotations, its role is equal to that of a client. [RQFuncBus2]

Negotiate about quotations
After the client received a quotation it would like to be able to send a counterproposal about the transport of the same goods with different time windows and/or costs. The client would like to receive a new quotation where the counterproposal is taken in consideration. If the transport company doesn't change its quotation then the client would like to receive a rejection on the counterproposal. If a new quotation is received then the previous quotation becomes invalid. [RQFuncBus3]

Create and change an order reservations
A client would like to make a reservation based on a quotation received from a transport company. As a response to the reservation, the client would like to receive a confirmation or a rejection. When the reservation is confirmed then it should include a moment in time when the reservation becomes a definitive order. Before this moment in time is reached the client has to be able to cancel the reservation or place an order based on it. The difference between a reservation and the period of validity of a quotation is that a reservation is a promise from the transport company while the period of validity is not. A client would also like to be able to change the content of a reservation, such as the addition of goods.

On the other side, the transport company would like to be able to receive reservations as well as changes to them. During the creation of a reservation, the transport company can make reservations for its resources required for the execution of the reservation. When a transport company receives a change requests for a reservation then it can try to change the reservation of its resources resulting in an confirmation of reject message to the client. [RQFuncBus4]

Place and receive orders
The client would like to change an order reservation to a definitive order. The client now knows that the transport company will execute the order. Analogue, the transport company would like to be able to follow this business process by changing the reservation into a definitive order. It then informs the about the acceptance of it. An reservation that is changed info an order can not be changed anymore. [RQFuncBus5]

Cancel reservations and orders
A client would like to be able to cancel a reservation. When the cancellation is sent before the moment in time that the reservation would become a definitive order then the client receives a cancellation confirmation from the transport company that canceled the reservation. A client would like to be able to cancel an order. Although the transport company can confirm the cancellation and stop the execution, the client cannot expect that the order isn't going to be invoiced. The client will receive a confirmation or a rejection of the request from the transport company. A rejection is sent when the transport company isn't able to stop the execution, for example when goods are already loaded on an airplane that is en route. A cancellation of an order that is already being executed is assumed to be an state that needs human intervention to solve the problem. [RQFuncBus6]

4.1.2 Information management

Transport company management
A client would like to manage a collection of transport companies that support the functionality of ELP. ELP therefore can be used to acquire quotations, place orders and keep track of executing orders. [RQFuncInf1]

Reservation and order management
A client would like to manage orders which are placed or are going to be placed at the transport companies that are mentioned in the previous requirement. A reservation is here assumed to be an order with a special state. The information about goods that need to be transported is used to

acquire quotations and place orders at transport companies. After placing an order, the information is used to keep track of the execution progress and to provide an order history. Analogue, a transport would also like to manage orders/reservations it received and placed (outsourcing). [RQFuncInf2]

Goods and transport schedule management
The client would like to be able to give information to the transport company about the physical aspects of goods that need to be transported. The information about the goods also contains information about the pick-up and delivery addresses as well as preferred time windows for pick-up and delivery. This information is required for acquiring quotations and placing orders. [RQFuncInf3]

Client and outsource management
A transport company would like to manage a collection of clients. These entries are used to link received orders to clients. Client information should contain information about addresses, such as settlement and invoice addresses, and about connectivity that can be used to reach the client as well as provide progress information of placed orders. A transport company can use other companies to (partly) outsource an order. The company would also like to be able to manage information about these companies. This requirement is analogue to the 'Transport company management' requirement. [RQFuncInf4]

Quotation management
A transport company would like to manage quotations that are sent to clients. These quotations can be used to create reservations or orders. Quotations include information about the client, goods, time windows, resources and quotation validity. [RQFuncInf5]

Resource management
A transport company would like to be able to manage its internal resources such as vehicles, warehouse space and employee availability. When the transport company knows which resources are used or available then it is able to create schedules as well as making decision about accepting, denying, outsourcing and executing orders. The resource information is also used to create quotations and make reservations of resources. [RQFuncInf6]

Split order management
A transport company would like to split the execution of an order into several parts which are assigned to internal and/or external resources. External resources are other transport companies that (a part of) the order is outsourced to. The transport company would like to know what resource is responsible for which part of the execution. Using this information, it is able to conclude that the complete transport is executed, i.e. the combination of the transport tracks, executed by the resources, starts at the original location and ends at the final destination. The complete track can contain transitions (unloading/loading/storing) of the goods from one resource to another. [RQFuncInf7]

Transport scale-up management
A transport company would like to be able to combine the transport of several goods together into the transport of one larger good. The transport company would like to know which goods are contained in other larger goods, such as sea containers or pallets. Using this information the company knows the locations of the contained goods by consulting the location information of the container. [RQFuncInf8]

4.1.3 Management information

Create management reports
Using the received quotations and placed orders at transport companies, the client would like to be able to derive information from the information system that can be used to create management

reports. These reports can provide insight into averages, totals, increases and decreases that are needed create decisions at management level. [RQFuncMan1]

Resource performance measurement
A transport company would like to use historical data to extract performance measurements. These measurements are important at the management level of the transport company. They should contain information about hours of usage, (exceeded) time windows, geographical information and costs. Historical data has to be available to create these measurement thus the data stored should include the necessary information. [RQFuncMan2]

4.1.4 Legal issues

Period of quotation validity
A client would like a quotation to have a period of validity wherein the client is able to place an order based on the quotation. The client can be sure that, when the order is placed within this period, it is reasonably accepted by the transport company, although it isn't a promise. [RQFuncLeg1]

Government specific
A transport company would like to have a collection of government specific legislation that can be used to execute orders without breaking a law. This information should also include customs information and documents that can be used for the import or export of goods. [RQFuncLeg2]

4.1.5 Track and Trace

Know where goods are located geographically
Clients and transport companies would like to know where goods are located geographically. These locations should be as accurate as possible, for example coordinates coming from GPS devices. When coordinates are not available in transportation means then they would at least know the geographical locations where goods have been last (un)loaded and stored. [RQFuncTra1]

Send and receive progress information
Progress information, such as the coordinates of goods described in the previous requirement, would like to be received by a client. This client can also be another transport company in case of outsourcing. This implies that a transport company that actually transports the goods has to send this information to its client. [RQFuncTra2]

Know where goods have been located geographically
As an addition the the previous requirement, a client and transport company would like to know where goods have been in the past. This information should include the (un)loading and storing locations although detailed locations of transportation means are not required. Using this information it is possible to provide a location track to the client that can be used for justification of the execution of an order. [RQFuncTra3]

Know who signed for completion
A client would like to know who signed for the Proof Of Pick-up (POP) and Proof Of Delivery (POD), both referred to as Proof Of Execution (POE), at the original location and the final destination. The POP transfers the responsibility of the goods to the transport company and the POD transfers the responsibility from the transport company to the receiver. As an addition to the POE at the original location and the final location, a client would like to know the POE at intermediate locations where the goods are (un)loaded and stored, for example because of outsourcing. [RQFuncTra4]

4.1.6 Functional requirements overview

The functional requirements described in the previous paragraphs are not all considered important. Chapter 3 has put the focus on the communication between transport companies when orders are outsourced what implies that management information or legal issues are out of scope. The following table puts a weight on every functional requirement and indicates which chapter focuses on which requirement.

Req. ID	Title	Importance	Chapter
RQFuncBus1	Requesting and providing quotes	Normal	6
RQFuncBus2	Get quotation from other transport companies	Normal	6
RQFuncBus3	Negotiate about orders	Low	Absent
RQFuncBus4	Create and change order reservations	Low	Absent
RQFuncBus5	Place and receive orders	Normal	6
RQFuncBus6	Cancel reservations and orders	Low	Absent
RQFuncInf1	Transport company management	Normal	7
RQFuncInf2	Reservation and order management	Normal	7
RQFuncInf3	Goods and transport schedule management	Normal	7
RQFuncInf4	Client and outsource management	Normal	7
RQFuncInf5	Quotation management	Low	Absent
RQFuncInf6	Resource management	Low	Absent
RQFuncInf7	Split order management	Normal	7
RQFuncInf8	Transport scale-up management	Normal	6,7
RQFuncMan1	Create management reports	Very low	Absent
RQFuncMan2	Resource performance measurement	Very low	Absent
RQFuncLeg1	Period of quotation validity	Very low	Absent
RQFuncLeg2	Government specific	Very low	Absent
RQFuncTra1	Know where goods are located geographically	High	7
RQFuncTra2	Send and receive progress information	High	6, 7
RQFuncTra3	Know where goods have been located geographically	High	7
RQFuncTra4	Know who signed for completion	High	7

Table 4.1 – overview of functional requirements

Functional requirements that are considered a low or very low importance are not described in this document and are considered future work. This doesn't imply that such a functional requirement is of no value. However, skipping them at first instance limits the scope and keeps the priority at requirements with a high importance. Chapter 6 focuses on business processes that are described by the functional requirements and optionally by existing solutions which are considered in the next paragraph. Chapter 7 focuses on a data model that can be used as a base to store and retrieve information that is used by the business processes described in chapter 6. From chapter 8 and further the focus is changed to more technical aspects that can be used to develop an information system to support the functional requirements.

◈ Q007

4.2 Existing solutions

Electronically exchanging information to automate business processes is not something new. Several solutions for exchanging information about business processes already are available. Exchanging information about business processes is generally referred to as Electronic Data Interchange (EDI). However, this term is also used as a reference to two specific standards of EDI, namely EDIFACT and ANSI X12, where EDIFACT has most users in Europe and ANSI X12 in the United States. This paragraph focuses on EDIFACT and several other EDI standards or designs and takes a closer look at whether they are suitable for the transportation industry.

Q101

To create a comparison between the existing solutions there are several questions that would like to be answered, namely:

- Is the existing solution an industrial standard?
- Does the existing solution support transportation business processes and/or custom businesses processes?
- What are the implementation costs?
- Can the existing solution be considered mature?
- What is known about document semantics?
- Are there any technical properties that attract attention in a positive or negative way?
- Are there any other properties that attract attention in a positive or negative way?

To answer these questions, the history, goals, details of communication and usage nowadays is considered. After considering the existing solutions, the answers to the questions are summarized and conclusion are made about which aspects of existing solutions are useful for ELP and which aren't. These conclusion are not only based on the functional requirements, but also on technical and general aspects that are seen in the existing solutions.

4.2.1 PapiNet

PapiNet [papiNet], which is an abbreviation for Paper Industry Network, originated from a group of European paper companies and some major German customers within the printing industry that decided to develop a business transaction standard using new XML technology in 1999. The reason for this organization to develop a new standard was to replace the now obsolete EDI by a standard that was cheaper to maintain and implement. The papiNet standard includes standard documents for purchase orders, shipping notices and invoices and can therefore be a possible solution for usage within the transport and courier industry branches.

Goal

The goal of papiNet is to enable companies, that are active within the paper and forest industry, to provide real-time exchange of information between buyers and sellers. PapiNet has developed standard electronic documents that are freely available to provide a “standard” for electronic information exchange for companies within the mentioned industry. The provided standard messages are meant to result in more structured processes with fewer data incompatibility issues. The first messages that were introduced by papiNet in 2001 are “Purchase order”, “Call off”, “Order confirmation”, “Delivery message” and “Invoice”. During the following years several other messages have been added, for example to retrieve product information and inventory status.

The real-time exchange of information using a “standard” is also mentioned in the research question. However, a major difference is the subject of the information that is exchanged. PapiNet primarily focuses on selling and buying specific products for the paper and forest industry while ELP primarily focuses on the ordering and outsourcing of transportation. The messages developed by papiNet clearly illustrate this, because most of them are specifically meant for a buyer to send to a seller or vice versa, such as the purchase of specific (industrial) products and the request for

inventory status. Ignoring these specific buyer/seller messages for physical products, several others remain that are meant for shipping and delivery.

Shipping messages

Two specific messages of papiNet focus on the transportation of the ordered products, namely “Shipping Instructions” and “Delivery Message” [papiNet-v230]. Both messages are sent to and from buyers and sellers as well as transportation partners. The scope of the Shipping Instructions message includes information about the products including quantities, requested delivery date and time, ship-to party, transportation means and the transport company. An examination of the Shipping Instructions message identified the data items about the sender and supposed receiving party as well as information about items that need to be transported, namely “Senderparty”, “ReceiverParty” and “ShippingInstructionsSummary”. However, the message also includes data items that are not of any importance for a transport company, such as “BuyerParty” and “SupplierParty” that provide information about the seller (or producer) and the legal entity to which the products are sold. The property of data items being mandatory or not creates a barrier for papiNet as a possible solution for communication between transport companies, because the data items BuyerParty and SupplierParty are mandatory while SenderParty and ReceiverParty are optional. This doesn't exclude that, if these data items are provided, the Shipping Instructions message is suitable for a transport company to know what goods need to be transported as well as the required from/to information.

◈ Q107

The other message that was examined is the Delivery Message. The scope of this message includes information about dates, such as shipping date, products with packaging and tracking details such as the route of delivery. The details of this message show several data items that are specific for the paper and forest industry, such as 'MillCharacteristics” that is meant for information about the mill party and machine that is involved in the production of the described product. The product data item in Delivery Message is mandatory while it is of no importance for transport companies. The predefined list of products made by papiNet also only contains industry specific products such as paper, pulp and recovered paper. In comparison to the Shipping Instructions message, this message contains more industry specific data items making it not very suitable for the transportation industry. However, a positive aspect of the Delivery Message is that it includes information about the delivery schedule as well as past deliveries during the complete transport (route of delivery). This collection of so called 'delivery legs' provides Track and Trace information about each item. Unfortunately, the documentation of papiNet doesn't mention any obligations to send this message. This results in a system that does support Track and Trace information, but no enforcement to keep the information up-to-date.

Maturity

PapiNet currently consists of group of more than 40 members in Europe and North America that participate in the development of the papiNet standard. The papiNet development team that was formed in 1999 is still active in 2008 with the latest release of the papiNet standard in spring 2008. During the past years more than 380 companies have implemented the papiNet standard at 830 sites and it can therefore be considered as a mature solution for information exchange about business processes within the paper and forest industry. Although considered mature, papiNet is not an ISO standard [ISO].

4.2.2 ELPIF

ELPIF [Zhang] is an abbreviation for an E-Logistics Processes Integration Framework that is based on web services. The idea for ELPIF originates from the fact that multiple (large) companies within the courier industry, such as United Parcel Service (UPS), Federal Express (FedEx) and Airborne Express (now part of DHL), all have their own interfaces to send purchase orders and retrieve Track and Trace information.

Goal
The goal of ELPIF is to define a common interface to communicate with companies within the shipping industry although it can be applied to other domains as well. ELPIF can be used to request quotations, to place orders and to keep track of shipments at multiple transport or courier companies while only having to communicate with one single web service. Although ELPIF is focused on clients communicating with transport companies, it is not excluded that the model can also be used for transport companies to communicate with each other when an order is outsourced.

Framework components
The framework incorporates three parts, namely the common alliance layer, the adaptation layer and a dynamic data binding mechanism. ELPIF is based on web services, because they are platform independent, easy to implement and all use XML that has advantages in the areas of data encoding and data formatting.

The first part of ELPIF, the Common Alliance Layer, defines a set of methods that every transport company has to support, creating an abstract high-level service interface publishing available services. If a transport company supports these methods then it can publish its web services at a UDDI registry [UDDI]. Although [Zhang] doesn't mention any specific UDDI registry it is assumed that all courier and transport companies use the same UDDI registry. Customers are now able to easily search for transport companies that can provide their services to them.

The second part of ELPIF, the Adaptation Layer, works as a service that operates between the published web service and the legacy system of the transport company. This layer manipulates the communication between the customer and the transport company in such a way that the legacy system communication interface doesn't need to be altered, which may cause incompatibility with other legacy systems and additional costs. In short, the adaptation layer converts every message between the customer and the transport company. The third part, dynamic data binding, takes care of replies consisting of live and updated data from the transport company.

Altogether, the three parts of ELPIF create a common interface, based on web services, to transport companies that can easily be discovered and enables customers to communicate with transport companies by only understanding this single set of web services.

Maturity
Although a search for ELPIF on the Internet has resulted in some documents referring to ELPIF, no implementation of it was found. One reason for this can be that the success ELPIF depends too heavily on the co-operation of (the mentioned) courier and transport companies to be feasible. Another reason for this can be that the document about ELPIF doesn't supply enough information or any specification to create an implementation of it. Probably the most important reason that no implementation was found is that the idea is patented by United States Patent 20030191677 [ELPIF-patent]. Q109

Despite the fact that ELPIF is nothing more than a patented idea on paper, this idea of one communication interface to communicate with transport companies is on the same wavelength as the research question. One specific subject that ELPIF mentions is the possibility to easily invite a quotation from multiple companies. It has to be pointed out that the patent can have consequences for ELP being implemented using web services and exported to the United States.

4.2.3 UN/EDIFACT

UN/EDIFACT [EDIFACT] is an abbreviation for United Nations/Electronic Data Interchange For Administration, Commerce and Transport. EDIFACT is one of the first initiatives to exchange electronic business documents and has been developed in the 1980s. EDIFACT has been

developed under the United Nations Economic Commission for Europe (UNECE).

History
One important historical aspect of EDIFACT is that, when introduced, a widespread digital communication network as the Internet was not available resulting in a situation where the electronic messages (or documents) had to be sent over private communication lines. Another historical aspect is the usage of information systems in general which were less used than nowadays due to the high costs. These two historical aspects made the use of EDIFACT, and EDI in general, only possible for large companies that had the resources to make the high investments for implementing EDI and the cost of communication [ShiwaFu]. A satisfactory return on investment (ROI) could be made by these large companies due to their cost savings on traditional communication such as fax messages that required many human resources to process them. Since small to medium sized companies would have a similar investment but a much lower ROI, due to the smaller amount of business documents sent and received, EDIFACT was not adopted by them except for those that were induced to do so by essential trading partners.

Goal
Before the existence of EDIFACT and EDI in general, business environments only used paper documents for their business activities. Exchanging paper-based documents had the disadvantage of extensive manual processes, manual intervention, interpretation and manipulation resulting in time delay, high labor costs and errors. The goal of EDIFACT was to create a standard for electronic documents that can be used for communication between trading partners. These electronic documents would solve all the mentioned disadvantages resulting in a paperless environment that required less human interaction and therefore save costs due to less human labor and errors. An even bigger benefit would come from the streamlined interaction between trading partners. This can increase inventory turns, decrease inventory, improve product and sales forecasting, decrease shipping costs, reduce product returns, improve cash flow and yield an improved relationship with trading partners [ShiwaFu].

The final character T in EDIFACT stands for Transport which implies that the collection of electronic business documents consists of those aimed for transportation. [Tedim-LDI] presents a list of 37 EDIFACT documents on the subject of transportation that take care of requesting quotations, order placement, cargo specifications, arrival notices and invoicing.

The ANSI X12 [ANSIX12] standard is the American counterpart of the in Europe developed EDIFACT standard. ANSI X12 and EDIFACT share many business documents although EDIFACT documents are generally longer and more complex than ANSI X12 documents. A result is that more ANSI X12 documents are required to exchange the same information as the number of EDI documents needed. There exist tables that map ANSI X12 documents to EDIFACT documents and vice versa; ANSI X12 is therefore not described in detail here.

Electronic documents
EDIFACT uses structured documents for exchanging business information that comply to strict syntax rules. These syntax rules are defined in such a way that the generated messages consist of only a few characters to identify message segments and to split the contained data elements into one or more components. An example of a segment to represent a persons full name and date of birth can be like “PRS+John:Smith+1970:10:20” (without the quotes) where “PRS” is the segment identifier, “John:Smith” is a data element with two components (first name, last name) and “1970:10:20” the date of birth data element consisting of three components (year, month and day). An EDIFACT document can consist of one or more (mandatory, conditional or optional) segments or groups where a group is a sequence of segments.

It can easily be concluded that EDIFACT messages consist of as little markup characters as possible. This results in EDIFACT messages that have a relative small size and are therefore

suitable to be transmitted over communication lines that were used during the introduction of EDIFACT. Small message sizes also suppress (very high) communication costs. Although an EDIFACT message has a clear advantage over XML when it comes to message size, especially when the message is big, it is considered an 'old' standard to format documents since it misses several advantages that the widespread use of the Internet and intensified globalization have introduced. Johan Koolwaaij of the Dutch Telematica Institute has defined a list of advantages and disadvantages of both (traditional) EDI and XML document formatting standards [Koolwaaij]:

Q505

EDI advantages:
- Efficient
- Mature B2B standards (EDIFACT and ANSI X12)
- Well known semantics

EDI disadvantages:
- Limited character set
- Overloaded and ambiguous
- No message validation
- High acceptance barrier

XML advantages:
- Unicode
- Simple, flexible, generic and extensible
- Widely supported by software on many platforms
- XML Schema's introduce data typing, reusable components, restrictions in syntax and defines relations

XML disadvantages:
- Only document formatting and no mature B2B standard (compared to EDIFACT)
- Big message size

Appendix B provides an example that illustrates the differences between an EDIFACT and equal XML message. It has to be emphasized that EDIFACT is a combination of a document formatting standard as well as a large number of predefined message structures, that can be used for a wide range of business transactions, where XML is only a document formatting standard. There is a lot of discussion about the future of EDIFACT and especially of using the EDIFACT semantics combined with the XML document formatting.

Maturity

Although EDIFACT was developed a long time ago, it is still maintained and used by many companies nowadays. The industries that use EDIFACT are mainly the automotive, civil aviation, tourism and retail industries. These industries have existed for a long time and have business units spread all over the world, which probably is one of the key factors for the success of EDIFACT in these industries. The benefits of electronically exchanging business documents are the increased speed within supply chain management as well as the elimination of language barriers that would exist when paper documents were exchanged. The high implementation costs, Internet as communication network, XML advantages, practical limitations and the decreasing number of EDIFACT specialists create an uncertain future for EDIFACT.

An industry that has not been mentioned is the transportation industry. This omission of EDIFACT and EDI in general is acknowledged by DHL Logbook that cooperates with the Technical University of Darmstadt [DHL-EDI]: "*Despite its many advantages, EDI is not widely used in logistics because of its high implementation costs. Instead, Internet-based variations are increasingly being used.*" Although EDIFACT is a mature standard for formatting and exchanging business documents with a long history and support for various business documents for transportation, it is not widely used by transport companies. This has motivated the initiative to design ELPIF.

4.2.4 ebXML

ebXML is an abbreviation for electronic business Extensible Markup Language [ebXML]. This international initiative was established by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for Advancement of Structured Information Standards (OASIS) [OASIS]. As a standard being established under the United Nations for electronic commerce it can be seen as the successor of EDIFACT. The development of ebXML started in 1999 and focused on five layers of substantive data specification for:

- Business processes
- Core data components
- Collaboration protocol agreements
- Messaging
- Registries and repositories

Goal

The goal of ebXML is defined by [ebXML-RS] as: "*provide an XML-based open technical framework to enable XML to be utilized in a consistent and uniform manner for the exchange of electronic business data in application to application, application to human and human to application environments*". ebXML would like to deliver technical specifications that consist of common XML semantics and related document structures to facilitate global trade and are internationally agreed on.

In contrast to papiNet and in accordance with EDIFACT, ebXML isn't targeted at only one business sector, but at every business sector from small and very big enterprises. This implies that the ebXML specifications and documents do not contain any specific information that can only be used within certain business sectors. This is where a contrast with EDIFACT appears, because EDIFACT does provide predefined document templates for specific businesses. Instead, ebXML provides core elements that can be used to define electronic business documents which then can be used within a specific business process. The first layer of data specifications provides a specification to describe business processes and the second layer provides the core elements. In short, ebXML provides specifications to describe business processes and electronic documents, but does not provide predefined versions of these documents.

Q203
Q204

Document exchange

As described in the introduction, ebXML does not provide any standard documents for predefined business processes. Instead, the first step to start using ebXML is to describe a business process that is a candidate for support by electronic data interchange. Describing the business process is done using the UN/CEFACT Modeling Methodology (UMM) [UMM] that utilizes a common set of *Business Information Objects* and *Core Components*. This methodology breaks business processes down into two views, namely the Business Operational View (BOV) and the Functional Service View (FSV) [ebXML-TAS]. These views are used to construct a *Business Process and Information Meta Model* (BPIMM) for an ebXML compliant application. The BOV is used to describe the semantics of business data and the architecture for business transactions. The FSV is used to describe the services for the mechanical needs of ebXML such as protocols and interfaces.

The next step is to publish a BPIMM using a registry service. This registry service serves as a storage facility for BPIMM, the used core components and a Collaboration Protocol Profile (CPP). A collaboration protocol profile consists of contact information, industry classification, supported business processes, interface requirements and messaging service requirements. After publication a companies' supported business processes can be discovered by other companies. If a company would like to conduct business then the CPP's of both companies are used to define a Collaboration Protocol Agreement (CPA) that consists of descriptions of the messaging service and business process requirements.

Before companies are able to communicate, they have to set up a messaging service that is used to send and receive messages (electronic documents) to and from other companies. Messages can be transported using various kinds of transport techniques such as SMTP and HTTP(S). The transport that is used is agreed on in the CPA. The CPA also contains information about authentication. After defining a CPA, the ebXML registry service is no longer required and companies can start exchanging messages about the published and agreed business processes.

Maturity
The specifications of ebXML have been published in 2001, which makes ebXML, just like papiNet, a relatively new standard for exchanging electronic business documents. Unfortunately, ebXML doesn't publish any information about how many companies are using implementations that are based on their specifications. The website of ebXML contains only a few case studies of which half are about companies or organizations that announced to use the ebXML specifications in their design. Searches for ebXML implementations on the Internet don't result in many positive results, because most of the results describe common information about ebXML. Another factor that can give information about the use of ebXML is the number of ebXML registries that publish information about business processes. Unfortunately, searching for these registries on the Internet doesn't give a positive result. However, one of the resulting websites, freebXML, contains a free implementation to start an ebXML registry [freebxml].

FreebXML is available for download at the popular open source software archive website Sourceforge.net. Searching this website for other implementations of ebXML specifications shows that most implementation projects do not have anything available for download and those that do, don't seem to be very popular. Most project started around 2001-2003 and until summer 2008 all projects together have had around 40.000 downloads of which almost 28.000 are downloads of freebXML that started in autumn 2001. Also, only 4 projects published files in the last two years.

The primary source for information about ebXML, the ebXML homepage, shows news items of which the most recent one was published in 2006. Also, the ebXML specifications have not been updated in the last years. The maturity of ebXML is concluded to be a bit contradictory. EbXML publishes a lot of information and specifications to design and implement software that uses the ebXML specifications for EDI. Despite this, the decisions to actually implement the ebXML specifications are not made or made in a negative way. The small number of ebXML implementations, registries and downloads of available software can only lead to the conclusion that ebXML cannot be considered a mature standard. The obsolete information on the ebXML homepage will probably also not work in favor of ebXML, because it is not likely that companies are willing to invest in adopting a standard of which the project team seems to be inactive. An exception is papiNet that ensures compatibility with ebXML and speaks about the introduction of ebXML core components in its FAQ [papiNet].

4.2.5 RosettaNet

RosettaNet is a non-profit organization that promotes electronic commerce by defining business processes, implementation frameworks and message guidelines [RosettaNet]. RosettaNet is named after the Rosetta Stone [RosettaStone] that was carved with the same message in three different languages, including hieroglyphs, that led to the understanding of hieroglyphs and translations. The name RosettaNet refers to the symbolism of understanding each other on the basis of standard processes for sharing business information between trading partners. RosettaNet was founded in 1998 by 40 leading IT organizations.

Goal

The standards specified by RosettaNet have been created to achieve the following goals:

- Define standard supply-chain transactions
- Standardize labels for elements like product descriptions and part numbers
- XML based business message schema's and process specifications
- Maximize reductions in cycle time, inventory costs, productivity and measurable supply chain ROI

The RosettaNet specifications primarily focus on business processes within supply-chain management. The descriptions of these business processes are defined as Partner Interface Processes (PIP) and are specialized system-to-system XML-base conversations. The RosettaNet PIPs are divided into seven specialized clusters to support business processes, namely [CoverPages-RN]:

1. RosettaNet Support: partner profile management
2. Product Information: detailed product information, product changes and technical specifications
3. Order Management: quotes, order entry, shipping, returns and finance
4. Inventory Management: inventory allocation, collaboration, replenishment, price protection and sales reporting
5. Marketing and support: lead and marketing campaign management, service
6. Service and support: warranty administration, technical service and support information
7. Manufacturing: transfer of design, configuration, process, quality and other manufacturing floor information

Document exchange

RosettaNet has described many business processes as PIPs that all have unique numbers according to the cluster they belong to [RosettaNet-CSP]. The clusters, and the PIPs belonging to a cluster, are divided into several segments. For example, the PIP for a purchase order at a supplier has PIP code 3A4, label “Request purchase order” and belongs to segment “3A: Quote and order entry” in cluster three. A PIP specification includes information about [RosettaNet-RNIF]:

- Partner business roles
- Business activities between the roles
- Type, content and sequence of business documents exchanged
- Time, security, authentication, and performance constraints of interactions

Every PIP specification consist of three major parts that have an overlap with those of ebXML. The three major parts are the BOV, FSV and Implementation Framework View (IFV). The IFV contains message guidelines such as data type and/or length of message elements. The structure and content of business documents exchanged is specified by XML Document Type Definitions that, together with the message guidelines, are used to validate documents.

Communication between trading partners takes place on a peer-to-peer basis where many different transport methods can be used, such as HTTP, FTP and SMTP. RosettaNet has support for the delivery of messages through hubs due to specific delivery headers. Business documents, that are XML-based, are encapsulated within a MIME structure. MIME enables the support for multiple message parts and encodings within a single message. A well-known application of MIME is within an e-mail message that contains the text message in both plain-text and HTML lay-out (alternatives within the MIME message) as well as an attachment (specific message part encoding). RosettaNet messages use MIME to split headers, such as a service and delivery header, and the business document. RosettaNet provides support for authentication, authorization and non-repudiation (digital signatures).

The collection of RosettaNet PIPs consists of a segment that is dedicated to transportation and distribution PIPs, namely segment 3B. This segment contains messages to place shipping orders as well as messages to change, confirm and cancel them. Other messages provide status inquiries, status updates and the communication of shipping documents. One aspect that attracts attention is that the initiative to send a message is always at the same participant, namely the shipper and not the shipping provider. This implies that status updates are not sent based on an event that occurred, but that the shipment status is requested by the shipper, based on polling. Another aspect that attracts attention is that the description of some of the PIPs purposes mentions the situation where "*the shipment is tendered to another Transport Service Provider at a gateway"*. Unfortunately, it is not made clear whether this is the final state of which a status message can be sent or that a shipping provider provides status request forwarding to this transport service provider. This also depends on whether the order at the next shipping provider is placed by the shipper or the previous shipping provider. Luckily, it cannot be excluded that RosettaNets shipping status inquiries can be used when an order is outsourced by a shipping provider.

One of the goals of RosettaNet is to define a standard for conducting electronic business for supply chain management. One of the consequences is that the collection of PIPs is generally meant for this goal. Unfortunately, it is not possible for a RosettaNet user to extend the collection of PIPs with custom PIPs to create support for (yet) unsupported business processes.

Altogether, the PIPs of RosettaNet that describe the supported business processes are mostly meant for supply chain management but also include a number of PIPs that can be used for order placement and track and trace at shipping providers which are, in this situation, courier and transport companies. They don't appear to be industry specific such as the messages used by papiNet and can be used in combination with outsourcing. Compared to other existing solutions, RosettaNet can be seen as an XML-based version of EDIFACT although the business documents within PIPs are not specifically equal to those of EDIFACT.

Maturity

Currently more than 500 companies worldwide, representing a trillion American dollars in revenues, actively participate in RosettaNet. Although the number of companies using EDIFACT is probably much higher, it exceeds the number of companies using papiNet and can be considered mature. To promote participants doing business with each other, RosattaNet provides a Trading Partner Directory on their website to search for other companies that have adopted its specifications in their EDI implementations.

Another aspect that illustrates the maturity of RosettaNet is the fact that Microsoft has adopted the specifications of RosettaNet into one of their product called BizTalk. This product is also made to conduct electronic business, but is not freely available and probably too expensive for most medium to small sized enterprises. BizTalk is primarily focused on communication between departments within a single company and is extended to external EDI with RosettaNet.

Not only Microsoft, but also many other large organizations use RosettaNet for their EDI implementations. An example is EDIFICE [EDIFICE] that is the European User Group for the Electronic Industry. This organization consists of European departments of large electronic enterprises such as Philips, IBM, Hitachi, Nokia and Siemens. EDIFICE supports UN/EDIFACT and, for XML, RosettaNet that is, according to their website, considered the industry standard.

4.2.6 Comparison of solutions

The existing solutions for conducting electronic business all have different properties that can be useful for ELP or not. The following table summarizes these properties:

Property	papiNet	ELPIF	UN/EDIFACT	ebXML	RosettaNet
A*	■■■■	■■■■■	■■■	■	■■■
B*	■■	n/a	■■■	n/a	■■■■
C*	■	n/a	■	■■■■■	■
D*	■■■	n/a	■■■■■	■■■■	■■■
E*	■■■■	n/a	■■■■■	■■■	■■■■
F*	■■■	n/a	■■■■■	n/a	■■■■
G	Yes	Yes	No	Yes	Yes
H	Extended data model to describe goods, track and trace support	n/a	Efficient messages, ISO standard, independent organization	ISO standard, independent organization	Support for many message transport methods (compatibility), transport outsourcing mentioned
I	Extended data model to describe goods too industry specific, no independent organization	Patented	Decreasing number of specialists	No electronic documents available	No independent organization

◈ *Q103 Q108*

** The score of this property is based on a scale of five: ■ - very low; ■■ - low; ■■■ - mediocre; ■■■■ - high; ■■■■■ - very high; n/a – not applicable (absent)*

A) Industry specific standard
B) Support for transportation business processes
C) Support for custom business processes
D) Financial barrier / implementation costs
E) Maturity
F) Document semantics
G) XML Advantages
H) General advantages
I) General disadvantages

Table 4.2 – properties of existing solutions

The table above can be used to identify positive and negative properties that can be used to make design decisions for ELP. It it also possible to take a look at properties that all available solutions have in common and that therefore are not considered as solution-specific properties. Similarly, one can look for properties that would like to be available but that are not provided by any existing solution.

One of the first decisions that can be made is that ELPIF has such high number of 'not applicable' scores that it can be ignored. Its high score on property A doesn't change this decision, because it is not of any value when most other properties are not applicable.

◈ *Q203*

First, the scores of properties A, B and C can be used to decide whether an existing solution is suitable for courier and transport companies. RosettaNet has the best support for this industry when looking at industry specific solutions. Its PIPs define useful and quite complete business processes when it comes to order placement and track and trace. The messages that are used in papiNets business processes are too focused on the paper and wood industry although its goods description specifications are suitable for the courier and transportation industry. EbXML has a very good score on not being industry specific, but this implies that all messages still have to be defined, and thus it provides only immature message semantics. EDIFACT is in between papiNet

and RosettaNet when it comes to being industry specific. Since RosettaNet, papiNet and EDIFACT all lack the support of support for custom business processes, the conclusion is made that RosettaNet is the most suitable existing solution when it comes to solutions that can be used straight away. Keeping ELP's supported business processes compatible with RosettaNets PIPs can increase acceptance and compatibility.

Q106

Secondly, the score of property D is examined. EDIFACT has the greatest financial barrier, because it has high implementation costs and requires expensive specialized knowledge. RosettaNet and papiNet have an equal barrier, although lower than that of EDIFACT, because they involve the implementation of an existing XML based solution. EbXML is also XML based, but requires business processes and messages to the be defined which involves extra costs. It is concluded that both papiNet and RosettaNet provide the least expensive solution and therefore the lowest the financial barrier. The financial barrier of ELP can be kept low if it is XML based and uses the applicable business processes and messages of papiNet and/or RosettaNet.

Q107
Q108

Thirdly, the scores of properties E, F and G are examined. EDIFACT has a long history which results in messages with good semantics although the introduction of new messages, which are based on existing messages, leads to ambiguity. Existing solutions that are based on XML can more easily be extended for future requirements. The lack of Unicode support of EDIFACT raises the question about how future proof this solution is and favors papiNet, ebXML and RosettaNet. The mediocre score of maturity and the lack of document semantics of ebXML leaves papiNet and RosettaNet as remaining solutions with the best scores on properties E, F and G. Although both are mature, the conclusion is made in favor of RosettaNet, because it has the most users and is supported by BizTalk.

Q507

Finally, the generic advantages and disadvantages of properties H and I are examined. The best possible solution would be an ISO standard that has a non industry specific data model, multiple transport methods for messages used for business process that are required by courier companies and doesn't consume too much communication resources. Unfortunately, this best solution is none of the existing solutions and the conclusion is made that the design of ELP should have as much of these advantages as possible.

Altogether, all of the existing solutions have disadvantages that make them not perfectly suitable for ELP. Appendix C provides an overview of functional requirements that are (not) supported by the existing solutions. Some of the requirements that are not met are described in the next paragraph. If a courier or transport company has to choose from the existing solutions then it can best decide to start using RosettaNet for electronic business, because this solution has the best overall score.

Q102
Q105
Q507

4.2.7 Requirements and design decisions

A requirement that is not met by any existing solutions is the possibility for real-time track and trace information (requirements RQFuncTra1 and RQFuncTra2). RosettaNet supports this partially when the status polling interval is set short enough, for example by introducing middleware that raises events based on polling frequently. The accuracy of information, that is used or known by a client and a transport company, in general is an aspect that none of the existing solutions really focus at. Two remarks can be made to this, namely first that accurate information uses many communication resources that were not available when EDIFACT was developed and second that a design based on ebXML can provide this functionality although it is yet non-existent. Providing real-time track and trace information based on events implies that a transport company 'tells' a client to update its information and thus that the client has to allow this.

Q506

Another aspect that existing solutions don't focus on is outsourcing. The delay of status information would increase if all transport companies are polling using some interval. This delay will be much

smaller if status updates are event-based which implies that all participants have to be allowed to change each others information until they have finished their part of the transport. In short, due to polling and the fact that each participant owns its own information, a rights management technique is not required.

Having considered several requirements that are not met by the existing solutions, there exist some functionality that they all have in common. All existing solutions are based on messages between two participants that are defined by a business process specification. All existing solutions have specifications of business processes to request quotes, place orders and request status information.

All the properties and aspects of existing solutions lead to the following design decisions for ELP, including references to solutions that create a foundation for it:

Q104
Q105

Nr.	Design decision	Design decision cause	Refers to existing solution(s)
1.	ELP supports, but should not be limited to, the following standard business processes: 'Request quote', 'Place order' and 'Provide status information'	These business processes appear within RosettaNet, EDIFACT and papiNet and will increase compatibility and acceptability.	RosettaNet, EDIFACT, papiNet
2.	It is possible for participants to agree on custom business processes	Flexibility that most existing solutions lack. Published custom businesses can more easily become standard business processes than proprietary third-party middleware solutions.	ebXML
3.	ELP uses messages between participants for business processes	A message can not be split, delivered using various transport methods and has predefined semantics.	All
4.	Messages are formatted using XML	XML provides many advantages, such as support for Unicode, and is widely accepted.	EDIFACT (disadvantages)
5.	Messages can be sent using several transport methods (e.g. SMTP, FTP, HTTP)	Various transport methods provide flexibility and lowers acceptance barriers.	RosettaNet
6.	Each participant has to provide a communication record that contains information how to communicate with this participant (protocols and parameters, authentication)	Ease of (initiating) communication and simplifying implementations due to a communication record standard that decreases the number of exceptions.	ebXML
7.	Track and trace information ('Provide status information') is event-based (not polling)	Using events saves communication resources and increases accuracy of track and trace information.	RosettaNet, papiNet
8.	Custom business processes can also use events	Increased business process speed and elimination of polling that wastes resources.	None
9.	Each participant should always have accurate information	Participants should be able to know whether their information is accurate to make justifiable decisions.	None
10.	ELP provides rights management to grant and revoke the possibility to change information	Allowing other participants to change one's information includes security risks that can be abused and thus have to be minimized.	None

Q204
Q506

Nr.	Design decision	Design decision cause	Refers to existing solution(s)
11.	ELP provides a data model for entities required by the standard business processes	Standard data structures are required for users to understand each others information.	PapiNet, RosettaNet
12.	ELP provides the possibility to extend the data model with custom extensions	Extending standard data structures with customs extensions increases flexibility and possible acceptance.	papiNet
13.	The data model supports outsourcing to one or more other participants	This enables companies to administrate the (parts of) orders that are outsourced and to communicate with a participant about only its part of the original order.	(RosettaNet)
14.	The data model supports transport up-scaling	The administration of transport up-scaling increases the accuracy of track and trace information about goods that are contained in another transport.	None
15.	Outsourcing is transparent to a client	A client holds a transport company accountable, but how the transport is executed is seen as confidential information of the transport company.	None

Table 4.3 – design decisions

Appendix C provides an overview of references between the design decisions and the requirements.

4.3 ELP design steps and document lay-out

The previous paragraph mentioned two major aspects that existing solutions don't focus at, namely accurate information for all participants and support for outsourcing between transportation companies. These two aspects are therefore the two primary aspects that this document focuses at, because the results provide information to extend existing solutions or a way to combine knowledge about existing solutions with aspects they are missing into a new (more transportation specific) solution.

The first step in the approach to design an information system that meets (most of) the functional requirements is to analyze the business processes of courier companies and transport companies that have been mentioned earlier. When the business processes are analyzed then it is possible to get a view on the primary entities and the flow of information between the different parts of these organizations and the external parties that are involved.

The next step is to extend the analyses of the business processes with the processes that are used when a company decides to outsource their orders instead of executing themselves. It is possible that companies outsource an order completely or only a part of it. It is also possible that an order is (partly) outsourced to more than one company where every company only performs a part of the execution. It can not be ruled out that an order is outsourced more than once.

A primary aspect of ELP is to provide a way to exchange information about orders when outsourcing is involved and every participant is informed about the progress. It should be possible to extract the flow of information between these companies and the client from the analyses of the business processes. These analyses have to provide the primary entities that can be used to

create a data model that describes them and their relations.

Business processes to a data model

As a result from the business process analysis it is possible to design a data model that describes the entities that occur in the order processing. Typical examples of entities that would probably be part of the data model are clients and orders. The attributes of the entities provide the information that is used to describe the necessary data that is needed to perform the business processes. Chapter 7 is dedicated to the design of this data model and the relationship between the entities.

One of the design decision of ELP is that it supports extendability. This implies that the data model should not be fixed without the possibility to add custom extensions. After defining the data model in chapter 6, it focuses on how the predefined data model can be extended to introduce flexibility and support for exceptions in standard supported business processes.

Having a data model that can be used for a single company including the flexibility to extend it, the focus is changed to the aspect of outsourcing where multiple participants are involved (exchanging information). The second part of the chapter describes outsourcing in relation to which parts of the data model are important and how they are used when an order is outsourced. It provides overview of steps and rules that have to be followed and obeyed to be sure that the correct information is exchanged and to create a view on the progress of an order by all participants involved.

Communication means to exchange information

Chapter 7 introduced a data model and an overview of participants exchanging information. However, companies that are exchanging information require communication means to be able to do this. Chapter 8 focuses on communication between companies. Some of the aspects that are attended are the actual reasons for companies to communicate, which companies have to communicate, what communication networks and topologies can be used and the increasing number of involved participants when an order is outsourced.

Techniques to provide accurate information

The result of chapter 7 and 8 is a data model to describes entities occurring in the business processes of chapter 6, the exchange of information and a communication network that can be used to exchange this information. Chapter 9 describes techniques that can be used for the actual exchange of information between participants using a communication network of chapter 8. This includes their ability to provide accurate information, and which of these techniques is the most suitable for ELP. Having accurate information is mentioned by design decision 9. One of the reasons for this is to be able to justify progress to clients. This means especially that, when an order is outsourced, the information about progress has to be generated by one participant and exchanged with the others using an information exchange technique. As an addition to techniques to exchange information, chapter 9 also focuses on rights management to change information that is of interest to more than one company.

ELP Prototype

Using the results of the previous chapters it should be possible to create a prototype that uses ELP to outsource orders and to provide accurate information about the progress to all the involved participants. Chapter 10 describes a prototype that has been designed using the business processes, data model and synchronization techniques.

Before describing the business processes, the next chapter covers the use cases which are used throughout the following chapters to compare solutions to real life situations.

4.4 Summary

To analyze existing solution and to be able to make design decisions for ELP, functional requirements are defined. These requirements are split info five categories, namely business processes, information management, management information, legal information and track & trace. To limit the scope of ELP, some requirements are partially or not considered, namely management information and legal information.

An analyses of the existing solutions is made based on the history, goals, details of communication and usage nowadays. The existing solutions that are considered are PapiNet, ELPIF, UN/EDIFACT, ebXML and RosettaNet. None of the existing solution is completely suitable for ELP, especially because they don't focus on outsourcing and providing accurate progress information. Therefore the design of ELP is especially focused on these two aspects. From the existing solutions PapiNet has the best overall score when these aspects are not considered important.

To create a design for ELP, the following steps are taken. First, a closer look is taken at the business processes in which clients and transportation companies are participants. Second, a common data model is designed that can be used as a model to represent information that is currently stored in proprietary information systems and is suitable for exchanging information when it comes to outsourcing. Next, to be able to exchange information, several alternative ways of creating a communication network are considered. Finally, having the business processes, the data model and a communication network, these are used to create a design of ELP that is closer to the application layer to exchange information.

5 Use cases to describe outsourcing

The business processes described in the previous chapter as well as all the technological aspects of ELP need to be compared to real life situations. In fact, the business processes describe these situations. This chapter describes use cases that can be used to check whether a designed solution would work with them. Another advantage of the use cases is the information that they provide about the flow of information and the flow of goods. The upcoming chapters of this document frequently refer to the uses cases, because this keeps these chapters close to the context of real life situations.

5.1 Goals and constraints

A use case consists of two parts that to describe a real life situation. First the participants of the use case are given. Second, the activities that happen are given. The activities that are performed by the participants, create a path for the flow of information between them. The activities and primary flow of information, such as order placing, are drawn in a scheme using a tree structure. The first property of the tree structure is that it describes how the order is outsourced. When the order is outsourced then the outsourcing company becomes a parent and the executing company becomes its child. The parent places an order at its child(ren) and sends the required data to the child need to execute it. An order scheme always starts with a client as root node, but it is not excluded that the client is another transport company or broker. In fact, when an order is outsourced, the outsourcing company is the client of another transport company.

The second property of the order tree is that the execution, that goes from the parent to the child, also carries over the responsibility. A child always has to justify the progress of an order to its parent. If this parent also has a parent then the justification is analogue, i.e. from the leaves to the root. When a company outsources its order to two or more other companies then this company is responsible for the whole order while the other companies are only responsible for their part of the execution.

Q004

A third property of the trees is the sequence of the order execution. The leaves, from the left to the right, are the companies that transport the goods in that order. A yellow transport shape indicates that the company doesn't outsource the entire order, but does a part of the execution itself. If the yellow transport is absent then this company outsources the entire order.

The following constraints can be defined for the order scheme:

- An order scheme always starts with a client as root node.
- A company can outsource the entire or a part of an order another company.
- When a entire order is outsourced then this company doesn't appear in the flow of goods scheme.
- When an order is outsourced then the outsourcing company places an order at the executing company and sends the required data that is needed to execute the transport.
- A company can do one or more parts of the transport itself and outsource the other part(s).
- The combination of transport that is done by a company together with the (partial) transport that it has outsourced is equal to the transport requested in the received order.
- A child always has to justify the progress to its parent.

Another scheme that is given, is the flow of goods. This information is used to get a sequence of all the resources that 'hold' the goods during the transport. The flow of goods can be seen as a flattened sequence of all the children of the order scheme. It therefore doesn't include all companies that are part of the order scheme. The order scheme doesn't give any information

about how the goods are exactly transported, for example, it doesn't give information about a possible temporary storage of the goods in a warehouse of a company; this information is provided by the scheme that describes the flow of goods. The current location of goods is a key aspect in 'Track&Trace' functionality.

The following constraints can be defined for the flow of goods scheme:

- The original location can not be the same as the final destination.
- The original location where the goods need to be picked-up doesn't have to be the same as the location of the root node (client).
- When an order is executed by more than one company then all the partial executions form a contiguous chain of transport from the original location to the final destination with possible intermediate storage.
- A flow of goods start with the transfer of goods from the original location to a transportation means.
- A flow of goods ends with the transfer of goods from a transportation means to the final location.
- The transfer of goods can be between two transportation means or between a transportation means and a non transportations means.

The order tree and the flow of goods are used as an important reference in the other chapters to design a solution that is conform these schemes as they describe real-life situations.

The following elements are used for the 'order' and 'flow of goods' schemes:

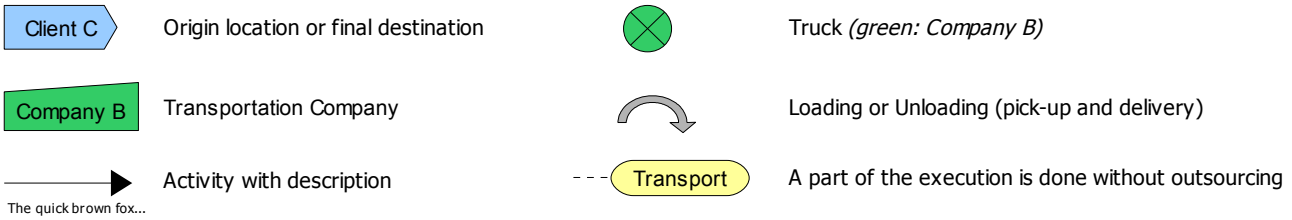


Figure 5.1 – elements used for order and flow of goods schemes

The color of the truck symbol identifies the company that owns the truck because that company has the same symbol color.

The final destination doesn't play an active role in the transport process in the 'flow of goods' schemes. It can be possible that this location is unaware of the fact that goods are sent to it, but it is assumed in the use cases that it will accept the goods and is therefore always the final shape in the 'flow of goods' schemes. The returning of undeliverable goods can be considered as a new order or as a change to the existing order. Although this should be possible with ELP it is out of the scope of the use cases.

☞ The use cases have trucks as transportation means, but these can be any other transportation means.

5.2 Use case 1: no outsourcing

Participants

Client C, transport company A (Company A), receiver company R (Receiver R)

Activities
Client C places an order at Company A. The order is about goods that need to be transported from Client C to Receiver R. Company A sends a truck to Client C to pick-up (load) the goods, drive to Receiver R and deliver them (unload).

Order scheme

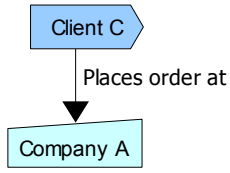


Figure 5.2 – order scheme of use case 1

Flow of goods scheme

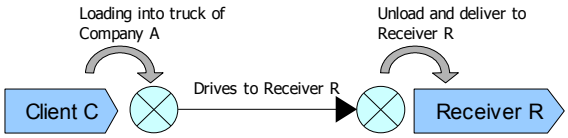


Figure 5.3 – flow of goods scheme of use case 1

5.3 Use case 2: single outsourcing

Participants
Client C, Warehouse X, transport company A and B (Company A and B) and receiver company R (Receiver R)

Activities
Client C places an order at Company A. The order is about goods that need to be transported from Warehouse X to Receiver R. Company A outsources the order partially to Company B. Company A sends a truck to Warehouse X to pick-up (load) the goods and drive to Company B where the goods are stored. Company B loads the goods into another truck and drives to Receiver R to deliver them (unload).

Order scheme

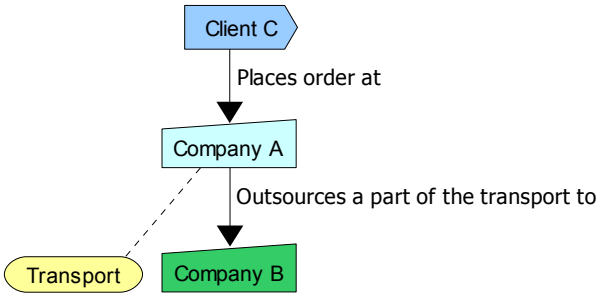


Figure 5.4 – order scheme of use case 2

Flow of goods scheme

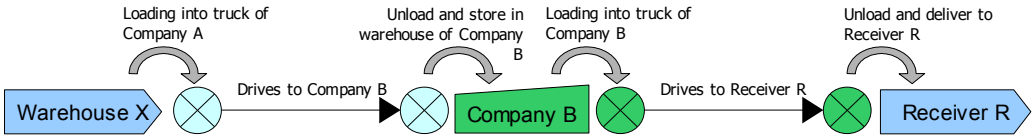


Figure 5.5 – flow of goods scheme of use case 2

5.4 Use case 3: multiple outsourcing, single level

Participants

Client C, transport company A, B and C (Company A, B and C respectively) and receiver company R (Receiver R)

Activities

Client C places an order at Company A. The order is about goods that need to be transported from Client C to Receiver R. Company A outsources a part of the order to Company B that has to pick-up (load) the goods at Client C and deliver (store) them at the warehouse of Company B. Company A outsources the other part of the transport to Company C. This company has to pick-up (load) the goods at Company B and deliver them at Receiver R.

Order scheme

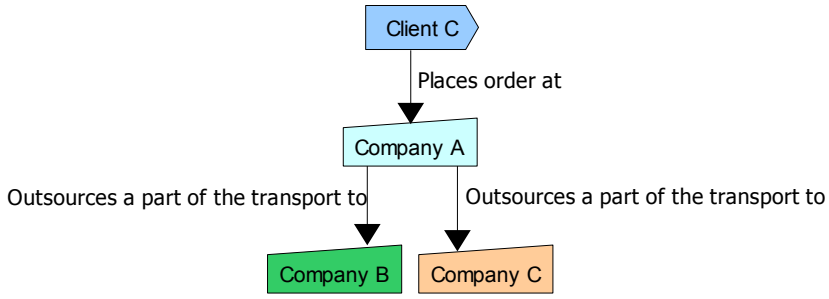


Figure 5.6 – order scheme of use case 3

Flow of goods scheme

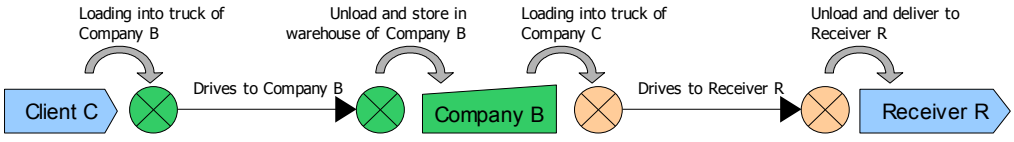


Figure 5.7 – flow of goods scheme of use case 3

5.5 Use case 4: multiple outsourcing, multiple levels

Participants

Client C, transport companies A, B, C, D and E (Company A, B, C, D and E respectively), receiver company R (Receiver R)

Activities
The activities are equal to use case 3 but with the following addictions. Company B outsources a part of its order to Company D. Company D has to pick-up (load) the goods at Client C and deliver (store) them at the warehouse of Company D. Company B outsources another part of its order to Company E that has to pick-up (load) the goods at Company D and deliver them at the warehouse of Company B.

Order scheme

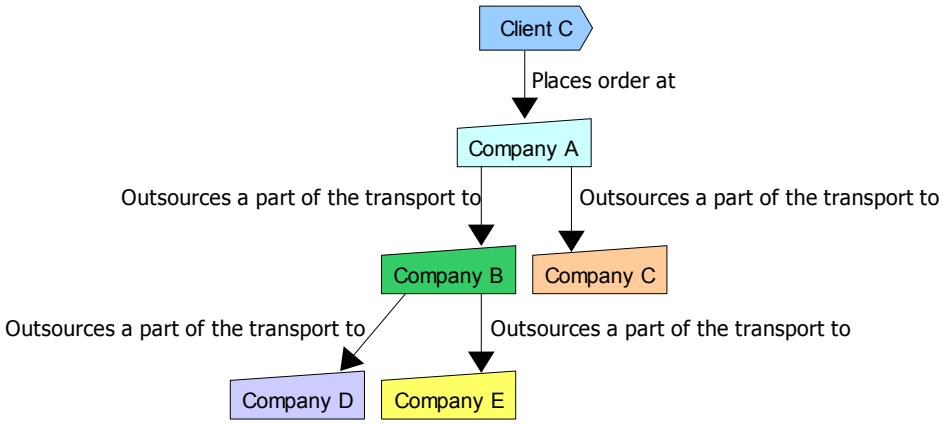


Figure 5.8 – order scheme of use case 4

Flow of goods scheme

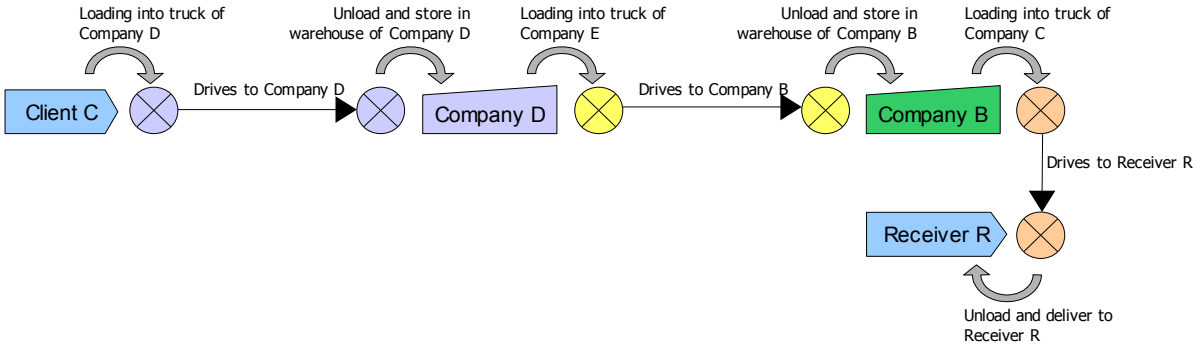


Figure 5.9 – flow of goods scheme of use case 4

5.6 Use case 5: outsourcing with many goods seen as one

Participants
Client C, transport companies A, B, C and D (Company A, B, C and D respectively), receiver company R (Receiver R)

Activities
Client C places an order at Company A. The order is about goods that need to be transported from Client C to Receiver R. Company A outsources the order to Company B. Company B transports the goods from Client C to its warehouse where they are stacked upon a pallet. Company B places an order at Company C to transport the pallet. The pallet is transported from the warehouse of Company B to that of Company D by Company C. Company D unstacks the goods from the pallet and delivers them at Receiver R.

Order schemes

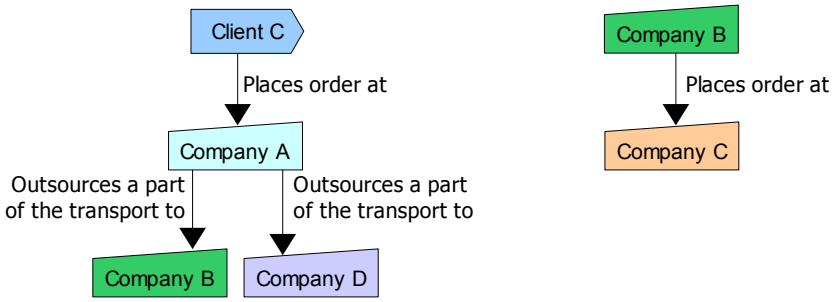


Figure 5.10 – order schemes of use case 5

Flow of goods scheme

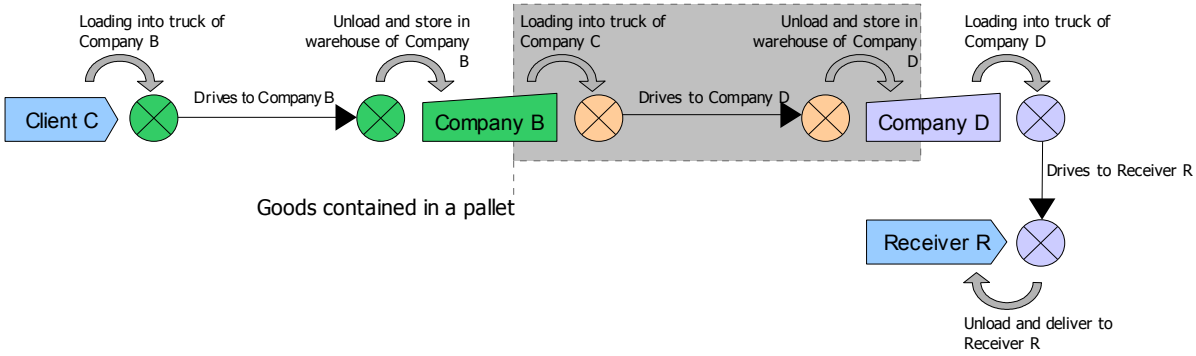


Figure 5.11 – flow of goods scheme of use case 5

It is clear that the use cases evolve from straightforward transport without outsourcing to a complex situation consisting of outsourcing together with transport up-scaling. The following chapter refer to the use cases to illustrate subjects that they are dedicated to.

6 Business processes analyses

6.1 Top level processes

The use cases described in the previous chapter are all combinations of business processes that are executed sequentially as well as partly in parallel. This chapter describes the business processes that are executed. The business processes are modeled using the Business Process Modeling Notation [BPMN] created by the Object Management Group [OMG]. The graphical elements that are used to create the models are described in appendix D.

The first paragraph gives an overview using a client that requests a quotation, also referred to as a quote, and places an order at a transport company. The order is executed and the progress is reported to the client. This Business Process Model (BPM) contains several collapsed sub-processes. These sub-processes are described in more detail in the next paragraphs.

The transport of an order involves two or more participants. First, there has to be a client that has goods that need to be transported. Second, there is at least one transport company that executes the transport. The client requests quotations from one or more transport companies and finally chooses one to place the order. This transport company can execute the order itself, but, as the use cases describe, can also outsource (parts of) the order to other transport companies. The BPM in figure 6.1 below describes the top-level business processes when no outsourcing is used.

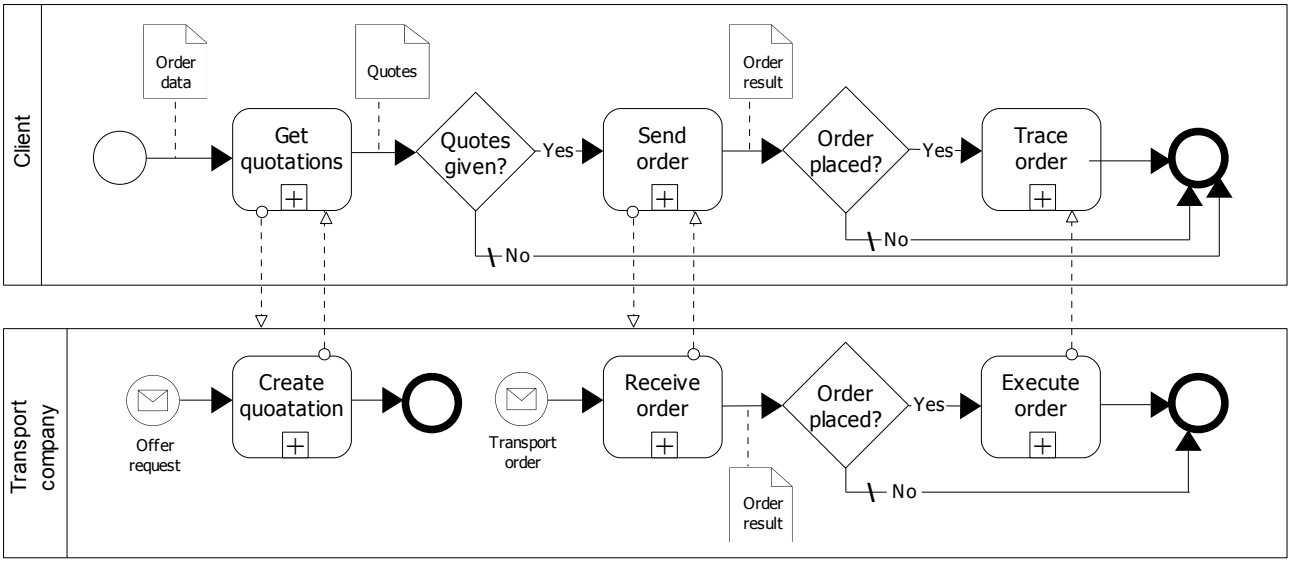


Figure 6.1 – BPM: top-level business processes

Figure 6.1 contains a Client pool and a Transport Company pool. The two pools together describe what business processes are executed at each participant and when messages are sent from one to another. This BPM doesn't explicitly include outsourcing although it does when some of the collapsed sub-processes are expanded in the next paragraphs. The ELP design decisions in chapter 4 mentioned three standard supported business processes. These business processes are represented by the combinations of 'Get quotations' and 'Create quotations', 'Send order' and 'Receive order', and 'Execute order' and 'Trace order'.

The business process that starts with goods to be transported at a client and finishes with all goods transported is referred to as the 'complete transport'. The model in figure 6.1 describes the

complete transport starting at the Start event. The transport company can't start the complete transport because it only has message events which are triggered by a message coming from the client. The top-level view of the complete transport is straight forward: first, the client has goods that need to be transported. It sends a quotation request to the transport company which triggers the 'Create quotation' business process at the transport company. Second, the transport company sends a quotation or a rejection back to the client. The model in figure 6.1 only has one pool for a transport company, but many of these pools can exists with quotation requests sent to each pool. When the client received a quotation it can place an order at the transport company. It is assumed that the client receives a quotation that satisfies its requirements.

The next collapsed sub-process in the clients pool sends an order, based on the received quotation, to a transport company. The transport company receives the order and sends a confirmation back to the client. The transport company now starts executing the order. The progress that is made, within the 'Execute order' collapsed sub-process is sent to the client. When the execution is finished then both client and transport company end the complete transport business process.

6.2 Client business processes

The next paragraphs will take a closer look at the collapsed sub-processes of the client and the transport company. First, the collapsed sub-processes of the client pool are described, followed by the collapsed sub-processes of the transport company pool.

6.2.1 Business process: get quotations

This paragraph focuses on the 'Get quotations' collapsed sub-process in the Client pool displayed in figure 6.1. The input of the sub-process are data objects that are used to create an order that is going to be placed at a transport company. The result of the sub-process is a list containing quotations from transport companies with, typically, time windows for the execution, prices, etc. The contents of a quotation is not defined as it would go beyond the scope of the business processes.

☞ *functional requirement RQFuncBus1*

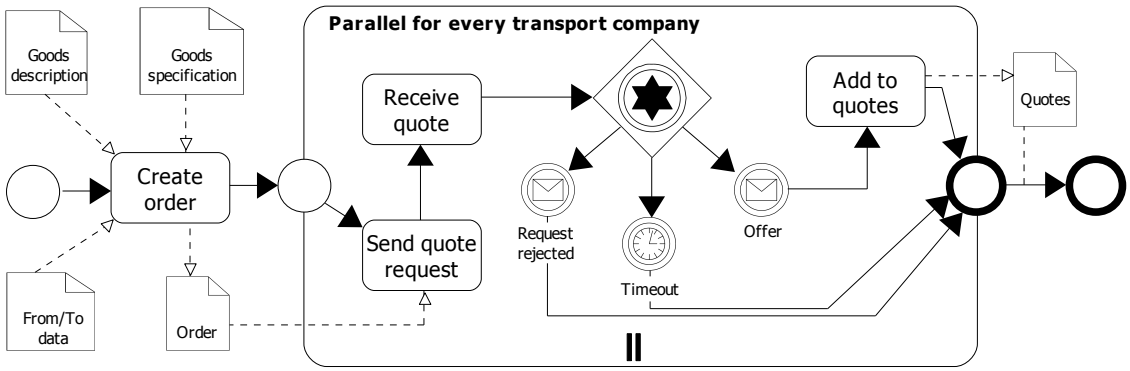


Figure 6.2 – BPM: get quotations

Figure 6.2 describes the BPM of 'Get quotations'. It starts with three data objects, namely 'Goods description', 'Goods specification' and 'From/To data', that are used to create an order data object. The order data object contains the data that a transport company needs to create a quotation for executing it. Using the order object, the client now is going to request quotations from transport

◈ Q003

companies it knows. The requests are executed in parallel for every transport company as indicated by the parallel activity. First, the client sends a quotation request to the transport company. Next, this company sends a message back to the client that handles it using the complex gateway. If the message contains a quotation then it is added to the list of quotations. Otherwise, by receiving a rejection or a timeout event, no quotation is put to the list of quotations and the instance of the parallel process finishes. Finally, the sub-process ends with a (empty or non-empty) list of quotations.

6.2.2 Business process: send order

The result of the 'Get quotations', described in the previous paragraph, is a list of quotations made by transport companies. The gateway in the Client pool after the collapsed sub-process decides whether to go on with the order or, if no quotation are available, end the complete process. When quotations are available then the next step[1] is to place an order at a transport company that is described in this paragraph. It is modeled by expanding the 'Send order' sub-process, see figure 6.1.

☞ *functional requirement RQFuncBus5*

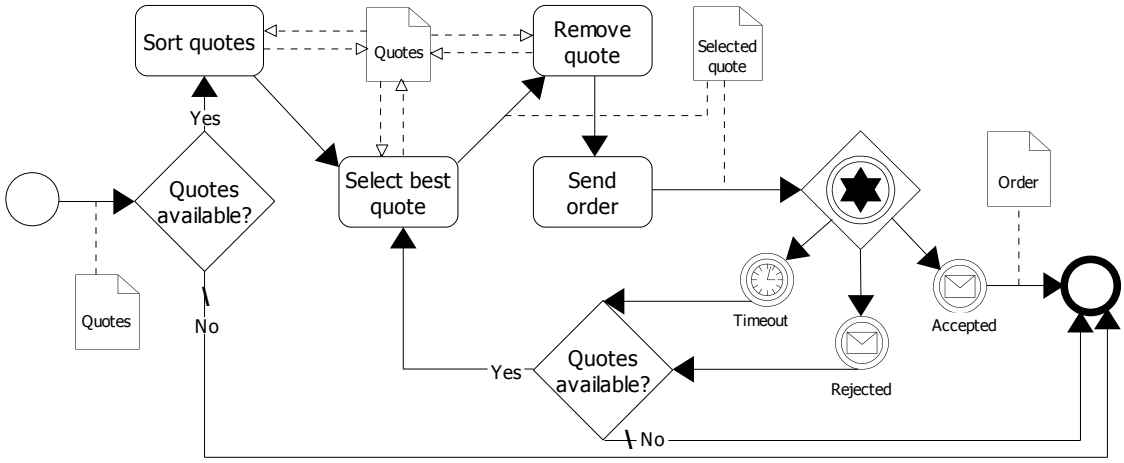


Figure 6.3 – BPM: send order

The 'Send order' sub-process starts with a list of quotations. When no quotations are given in the previous 'Get quotations' sub-process then this list will be empty and, due to the first gateway, this sub-process ends without a a placed order. When the list of quotations is not empty then the quotations in the list can be sorted according to the specific demands of the client. The client now selects the best quotation and removes it from the list. The best quotation is used to send an order request, that refers to the quotation, to the transport company. The order request will be received by the 'Receive order' sub-process of the transport company. This sub-process, which is described later, sends a 'rejected' or 'accepted' message back to the client.

When no message is received from the transport company after a certain amount of time then a timeout occurs and the order is processed as being rejected. The rejection and timeout event are followed by a gateway that determines whether more quotations are available, in which case the sub-process is repeated from the activity where the best quotation is chosen. Without any alternative quotations the sub-process ends without a placed order.

When the client received a 'accepted' message then the order, based on the quotation, is placed.

1 Creating a reservation based on a quotation is skipped due to the absence RQFuncBus4

Figure 6.1 tells that the transport company starts executing the order and the client is tracing its order. This is the sub-process that is expanded in the next paragraph.

6.2.3 Business process: trace order

The 'trace order' sub-process is based on three events that can occur using a complex gateway. One event is a 'progress' message received by the client. This message is sent by the transport company and informs the client about progress that is made during the execution of an order. The client updates the order data and waits for the next event to occur.

Another event that can occur is receiving a 'finished' message from the transport company. This message informs the client about the order that is completely executed. Typical data that is included within this message is a Proof Of Delivery (POD). The client knows that the order is finished and the trace sub-process also finishes.

A third kind of event that can occur is a timeout. This event can occur when the time passed, since the start of the execution, is beyond reasonable and the order should be finished already. The timeout events ends the sub-process with an error state.

☞ *functional requirement RQFuncTra2*

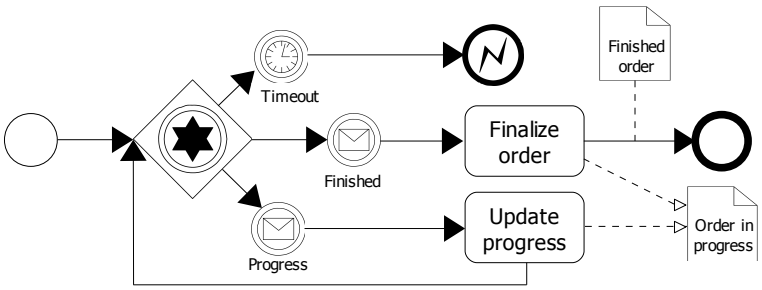


Figure 6.4 – BPM: trace order

6.3 Transport company business processes

The following paragraphs focus on the sub-processes of the transport company pool of figure 6.1. These sub-processes also include business processes that are used for outsourcing from one transport company to one or more others. When a transport company outsources (a part of) an order then, in fact, it becomes a client of another transport company. The transport company uses the 'Get quotations' sub-process of the clients pool to get quotations from companies that it can outsource (a part of) the order to. The next paragraphs will focus on the sub-processes that are part of the pool of the transport company.

6.3.1 Business process: create quotation

The 'create quotation' business process handles the event of a quotation request from a client. The model in figure 6.1 illustrates that a client sends a quotation request to a transport company containing data that is needed to create a quotation. The transport company tries to create a quotation, described in this paragraph, and finally sends either the quotation or a rejection. Figure 6.5 contains the business process model of the expanded sub-process 'create quotation'.

☞ functional requirements RQFuncBus1, RQFuncBus2

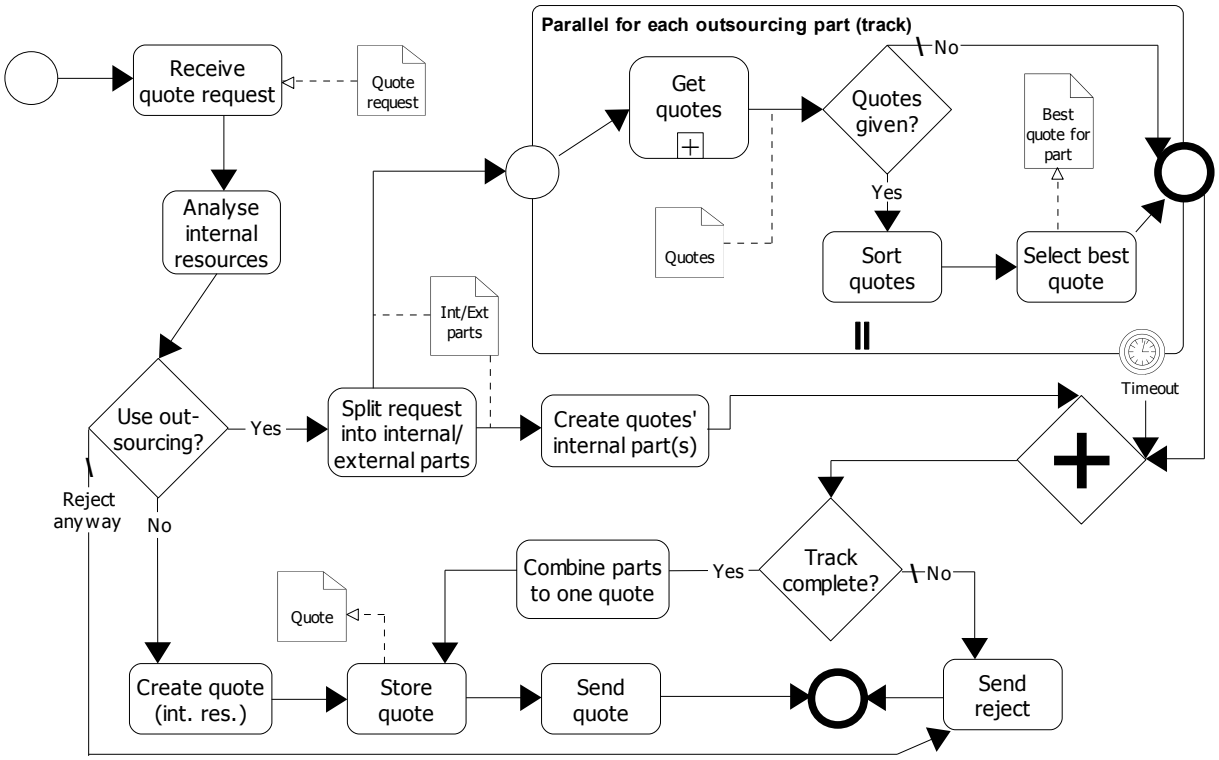


Figure 6.5 – BPM: create quotation

The transport company first receives the quotation request and analyses its internal resources. Internal resources are, for example, the trucks, employees and warehouse facilities that belong to the transport company. Using the analyses, the company is able to decide whether it can execute the request using internal resources or (completely) using outsourcing. Another possibility is to reject the request anyway. The decision to use outsourcing is made in the first gateway. When no outsourcing is used then the quotation is based on the use of internal resources. The created quotation is stored and sent to the client.

When the company decides to use outsourcing to create a quotation then it is possible that it fully outsources the execution of the request or only a part of it. The company splits the request into parts that are either executed by internal resources or by outsourcing. All parts together have to form the transport from the original location to the final destination. After the creation of these parts, the model forks the flow into the creation of quotations for the internal and external parts. It is possible that the request is fully outsourced in which case no internal parts exist. The creation of a quotation for these parts is then simply skipped and the flow goes immediately to the join gateway.

As the company decides to use outsourcing, there is at least one part that is executed by another transport company. The company would like to receive quotations for every part of the request that it would like to outsource. These quotations are requested in parallel within the multiple instance task object. First, quotations for the part are requested within the 'Get quotations' sub-process that is described earlier in this chapter. The next step of the multiple instance task is to select the best quotation for the part which is analogue to the selection of the company to place the order within the 'Send order' sub-process of the clients pool. Finally, when a quotation is selected for each part, then the flow continuous from the join gateway. This can also happen when it took to long time to receive and/or select a best quotation for a part in which case the instance of the multiple instance task reaches a timeout.

Within the next gateway, after the two flows are joined, the quotations for all internal and external parts are checked to see whether the concatenation of them is equal to the transport track that is requested by the client. If it is equal then the quotations for the parts are combined together in one quotation that is stored and sent to the client. If it is not equal then the company wasn't able to create a suitable quotation and sends a reject message.

A remark that can be made to this expanded sub-process is that it introduces recursive calls to its own model. The 'Get quotations' sub-process in the multiple instance task starts the 'Create quotations' sub-process at each transport company it would like a quotation from. As this sub-process can start its own 'Get quotations' sub-processes it can even be possible that transport company A, that requested a quotation from company B, receives a quotation request from company B to execute the same transport (circular quotation request). A solution to this problem can be to uniquely identify the goods and to assume that this identifier is contained unchanged within a quotation request. A company can now easily check whether it already has the goods in a 'request quotation state' and reject the request anyway.

There is no limit on the number of times that an order is outsourced although it is assumed that it is limited due to practical limits such as profit margins that all involved companies would like to have.

6.3.2 Business process: receive order

The pool of the transport company in figure 6.1 contains collapsed sub-process 'Receive order' that it initiated when a 'Transport order' message is received. The result of this sub-process is that the order is placed or that it is not placed, including the parts that can be outsourced. Figure 6.6 below illustrates the expanded sub-process of 'Receive order'.

☞ *functional requirement RQFuncBus5*

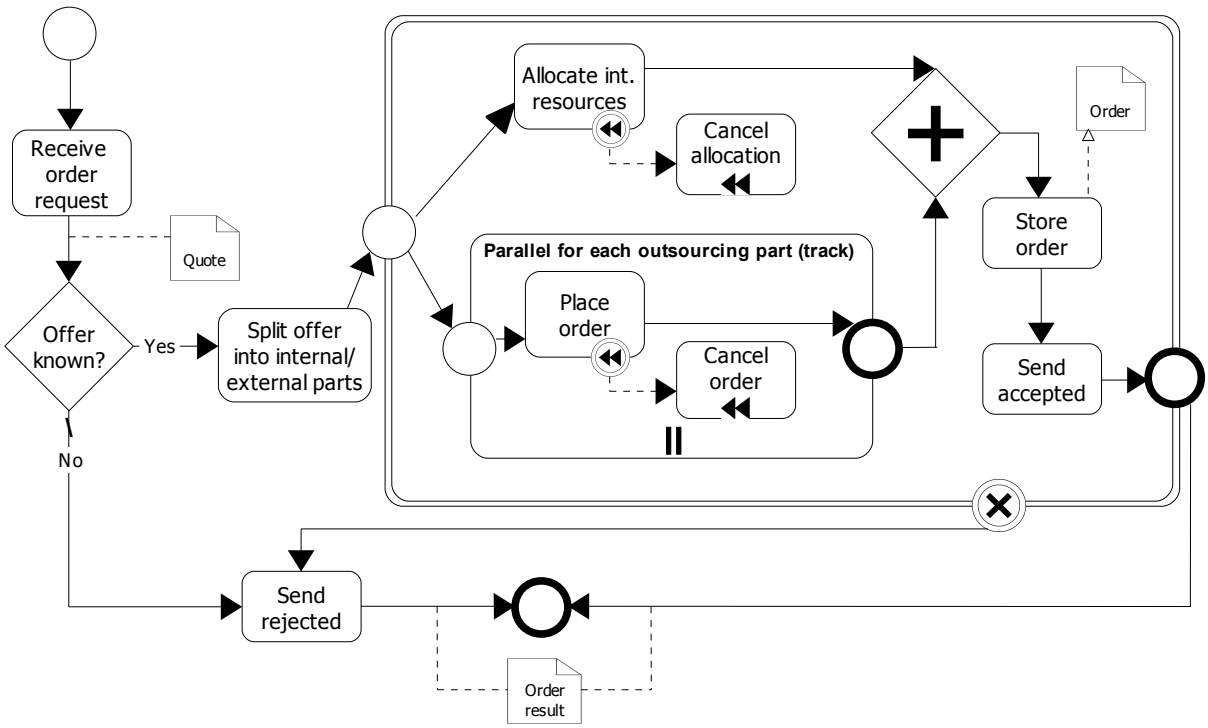


Figure 6.6 – BPM: receive order

When the transport company receives the order requests, that includes a reference to a quotation, then it checks whether the quotation is known. If the quotation is not known then the company rejects the order immediately. When the quotation is known then it is split into internal and external parts and an order placing transaction is started.

The transaction forks the flow into one an internal and external part. If there is no internal or external part then the activities in the flow are skipped. The company tries to allocate the internal resources that are part of the quotation. If the resources cannot be allocated, for example because they are already allocated for another order, then the transaction is canceled and every allocation/ order placement is rolled back. When the resources can be allocated then the flow waits for the completion of the outsourcing flow at the gateway.

The quotation given contains references to the quotations from transport companies that are used for the outsourcing of the order. The company now tries to place the orders for the outsourced parts. If one part cannot be placed then all others, as well as allocated internal resources, are canceled.

When both the allocation of the internal resources as well as the orders for the outsourced parts succeeded then the order is stored and a confirmation is sent to the client. The transaction now commits. When the allocation of the internal resources or an order of an outsourced part failed then the transaction is rolled back and a rejection message is sent to the client.

To improve the success rate of the transaction, a quotation can include a period in which the quotation acts as an option on the resources. The period shouldn't be too long, but long enough for a usual execution time of the 'Get quotations' sub-process. When the order is successfully placed, and probably an option is used, the client must cancel the other options to release the reserved resources at the other transport companies.

6.3.3 Business process: execute order

The last sub-process in the pool of the transport company in figure 6.1 is 'Execute order'. This business process is only started when an order was successfully placed by the client. The result of this business process is an executed order including information on the progress that was made during the execution. The sub-processes is expanded in figure 6.7 below.

☞ *functional requirement RQFuncTra2*

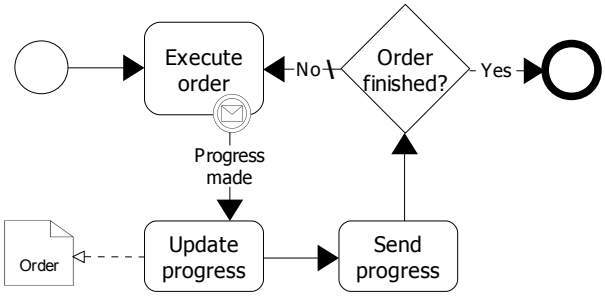


Figure 6.7 – BPM: execute order

The process starts with the execution of the order. At some moment in time there will be some progress, for example the pick-up of the goods. An employee of the transport company will send a message with the progress that is made. This progress is stored and is sent to the client. This will trigger the 'Progress' event in the 'Trace order' sub-process. When the progress message isn't indicating that the execution is finished then the transport company will continue executing until the

next event occurs. Finally a 'Finished' message will be received and this message is also sent to the client. Both the client and the transport company will now finish the order and therefore the complete transport is finished.

When an order is outsourced by a transport company then it starts the 'Execute order' sub-process beginning with the 'Execute order' activity. Assuming that the complete order is outsourced, the company doesn't execute anything itself, but waits for messages from the transport company/companies that the order is outsourced to. When a message arrives then it updates its own order data and sends the progress to the client. The 'Send progress' activity of the transport company where the order is outsourced to doesn't send the progress message to the client, but sends the progress message to transport company that outsourced its order. In short, when outsourcing is used and the actual executing company makes progress then a chain of progress messages will flow from that company back to the client, updating everyones order data. This chain is equal to use case trees going from a leaf up to the root.

6.4 Use cases and BPM

The use cases described in chapter 5 should be instances of flows in the business process models of the previous chapters. This paragraph focuses on each use case to check whether these instances can be made using the business process models.

6.4.1 Use case 1: no outsourcing

The first use case consists of a client that places an order at a transport company. It is executed by that company. First, the client requests quotations from one or more transport companies within the 'Get quotations' sub-process. Next, it selects a quotation and sends an order to Company A. This company accepts the order and executes it. This use case fits exactly in the BPM described by figure 6.1.

The order tree of use case 1 is build using two phases that are not described in the use cases chapter. The use cases don't give any information about quotations that are requested and only display the companies that are involved when an order is placed/outsourced. The order tree of use case one is built by going trough the following stages that are derived from the business processes of this chapter:

Initial state: a client with goods that need to be transported
Quotation stage: the client requests quotations from transport companies
Order stage: the client places an order at one of the transport companies
Execution stage: the transport company executes the order
Final state: the order is executed.

All financial aspects such as invoices and payments are not taken inconsideration as described in paragraph 6.2.1.

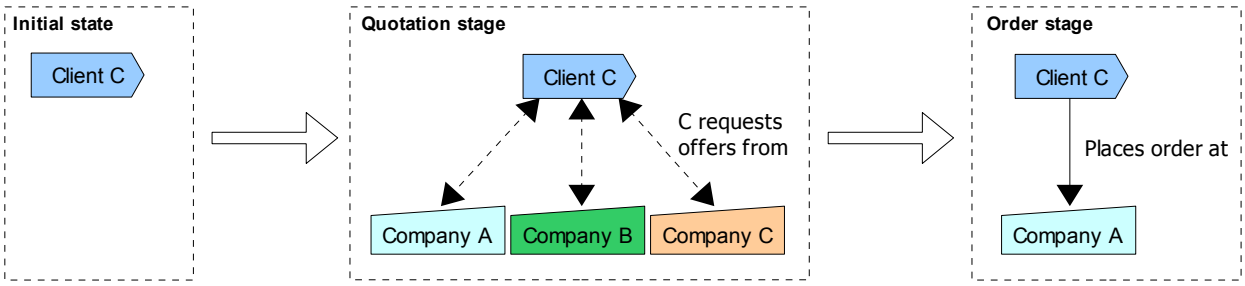


Figure 6.8 – two steps from initial state to order stage

Figure 6.8 above describes two steps, from the initial state to the order stage, that result in the order tree displayed of use case 1. Client C requests quotations from several companies (quotation stage) and decides to place an order at company A (order stage). The (successful) result of the order stage is displayed in the order tree of the use case.

The execution stage is also not mentioned in the order trees of the use cases. From sub-process 'Execute order' and 'Trace order' it can be concluded the initiator of messages sent changes from the client to the transport company. These messages form the justification from a child to its parent described in paragraph 5.1. The execution stage can therefore also be drawn as the order tree except that all the arrows are turned around and form the justification using progress messages. The final state is equal to the initial state except that the goods are transported.

6.4.2 Use case 2: single outsourcing

Use case 2 starts with the same sub-process as each other use case, namely 'Get quotations'. This invokes the 'Create quotations' sub-process of transport company A. The sub-process splits the order into an internal part and an external part. For the external part, the transport company starts the 'Get quotations' sub-process itself and requests a quotation for the part it is outsourcing. The quotation stage displayed in the previous chapter would be drawn with an extension of several transport companies as children of company A and probably children of the other transport companies if they would also use outsourcing for the order. Company A decides that the quotation of Company B is the best one and uses it together with the quotation of its internal part to create the quotation for the client. The clients decides to use this quotation (sub-process 'Place order') which results in the order tree of use case 2 when the sequence flow reaches the start of the transaction of sub-process 'Receive order'.

In paragraph 6.3.1 it is pointed out that the 'Create quotation' sub-process creates recursion within the complete process. When this recursion is suitable within a single outsourcing use case then it is also suitable for use cases that use multiple outsourcing and/or multiple levels since these use cases only create more instances of the 'Create quotation' sub-process. Use case 3 and 4 can therefore also be described using a sequence flow of the business process models.

6.4.3 Use case 5: outsourcing with many goods seen as one

Use case 5 contains the scenario where the goods are loaded upon a pallet that is transported from Company B to Company D. The transport of this pallet is a new order. Company B is responsible for the transport of the goods from its warehouse to the warehouse of Company D. Usually company B would load, transport and unload the goods and send this progress to Company A. In this use case these activities are performed within the new order that company B places at company C. Since company C is unaware of the goods stacked on the pallet, it doesn't send progress messages for the goods of Client C to Company B or A. However, it does send progress messages about the execution of the new order to company B. Company B is the only company that knows that the goods of Client C are stacked on the pallet and therefore Company B, that receives progress messages of the pallet transport, has to use these messages to update the progress of the transport of the goods. In short, when the transport of the pallet makes progress then the transport of the goods makes progress also.

The business process models described in this chapter do not have any activities involved with combined transports with, for example, a pallet. These activities are added to the existing models described in the previous paragraphs.

The first activity that has to be added is when goods are combined into 'new goods'. This activity

has to store information about what goods are contained in the new goods. In the use case, this would enable Company B to detect that the goods of Client C make progress when the transport of the pallet makes progress. This small business process, that can be called 'Combine goods', with one ore more data of goods as input and one data of goods as output is not modeled here.

The second activity has to be added to the sub-process 'Execute order'. This activity checks for every progress message whether the goods that are described within the message contain other goods. If so, then the transport company needs to send a progress message the other sub-process 'Execute order' that is about the goods that are contained. The altered sub-process 'Execute order' displayed below.

☞ *functional requirements RQFuncTra2 and RQFuncInf8*

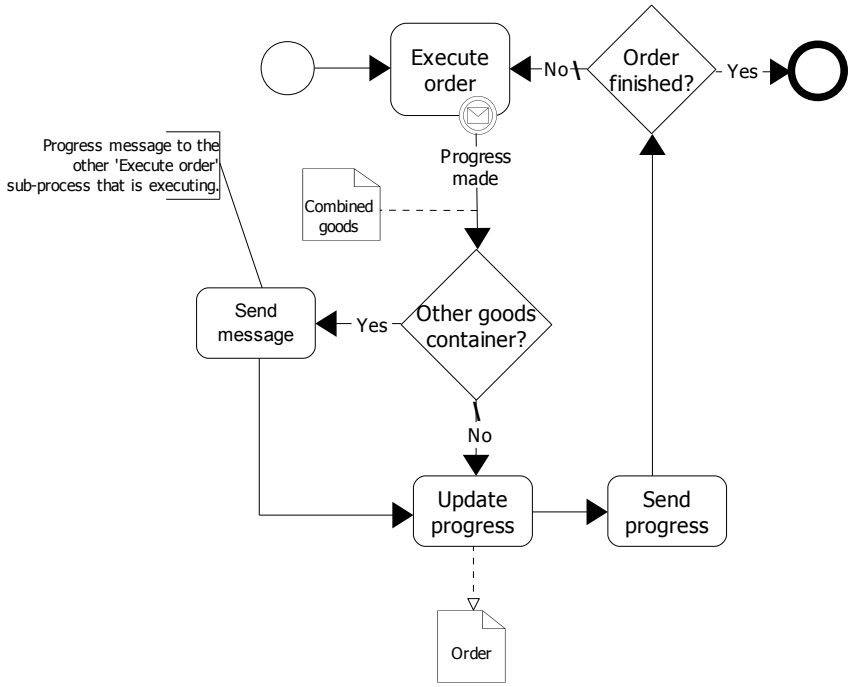


Figure 6.9 – altered BPM: execute order

Company B is executing a transport of the goods of Client C (BP-A). It is also involved in the execution of the transport of the pallet (BP-B). When a progress message of the pallet transport arrives at company B then the gateway checks whether there are other goods contained by this transport. This is true in this use case, thus the company sends a (internal) message to the business process BP-A. This business process will also check whether it contains other goods, but this is false. Both processes will update their progress and send progress messages to the participant it received the order from.

The solution for combined transports only works if Company B places an order for the transport of the pallet at itself and outsourced it to Company C. When Company B would only acts as a client of company C then it wouldn't have sub-process 'Execute order' active and therefore wouldn't send the progress message to BP-A.

6.5 Crossing company borders

One of the major differences between a company that accepts and executes an order and one that accepts and outsources an order is that the latter involves a more complex situation because a lot of business aspects cross its company border. The previous paragraphs have shown a superficial view on outsourcing that exists of simply acquiring quotations, placing orders and receiving progress updates. Although this superficial view corresponds with real life situations, this paragraph focuses in more detail on aspects that appear when business processes cross a companies' border.

The first aspect is the introduction and shifts of responsibilities and accountability. A company that outsources an order keeps all responsibility and accountability to its client, but it has to realize that to it becomes dependable on another company when it is actually held accountable. This implies that the outsourcing company has to realize that, by outsourcing the order, its influence on a successful execution decreases although its responsibility and accountability remain equal. Realizing this, an outsourcing company shall not only make its decision, who to outsource the order to, based on the lowest quotation, but also on a companies reputation and successful cooperation in the past.

Another aspect is the agreement on obligations and requirements that participants need to have. Many obligation and requirements can be thought of, for example:

- Agree on the contents of an order and a companies flexibility to deviate from it.
- What progress information needs to be provided and what requirements are there on the frequency and/or accuracy of this information?
- What are the requirements for exchanging information? For example, is information exchanged between humans (voice, paper), information systems (EDI) or a combination of these and which language (protocol) is used as well as the informations' level of detail?
- Who is responsible for the information and updates of this information? This has influence on the possibility to hold someone accountable in case information is incorrect.
- Are there any restrictions on which company an order can be outsourced to? For example, company A doesn't like to do business with company X. However, A outsourced its order to company B, that in its turn does like to do business with and outsource its order to company X.
- What are the requirements for confidentiality?

In general, companies that do business need to agree on many aspects that can be divided into the categories operational, financial, legal and technological. If no agreement can be made then this can cause unforeseen problems that are not likely to exist in case of no outsourcing. The categories operational, financial and legal are assumed to be out of scope of this document, although it is worth emphasizing them to exist. The next chapters, that focus on more technological subjects, do keep them in mind to provide solutions where possible.

◈ Q302

6.6 Summary

The functional requirements introduced several business processes that would like to supported by ELP. The business processes are steps performed at a client and a transport company starting with quotations and resulting in an executed order of which the progress can be observed by the client. The following business processes are analyzed and modeled in detail: get quotations, create quotations, send order, receive order, execute order and trace order. The business processes provide a view on the messages that are sent between a client and a transport company. Using the use cases, these business processes are extended with steps that are taken when an order is outsourced. Outsourcing introduces many aspects that a pair of outsourcing and accepting companies need to agree on to prevent conflicts and disappointments. These aspects can be divided into the categories operational, financial, legal and technological of which this document primarily focuses on the technological aspect that can support the other categories.

7 Data structures as support for business processes

7.1 Introduction to a common data model

The business processes described in the previous chapter use several kind of data objects and messages. An order is modeled as a message that is sent from the client to the transport company and contains order data. Equal messages are used when a transport company outsources an order to another company. In practice, these data objects and messages have to be created and understood by existing information systems. The data objects that, for example, contain orders and goods need to be described by a Common Data Model (CDM) that is supported by the existing information systems. It doesn't need to be the model of the data of the software itself, but the information systems need to have at least a two-way mapping function to read and write data about a transport from and to the common data model.

The difficulty with a common data model will always be that it isn't completely suited for all systems that are used nowadays. To maximize the usability and compatibility with current information systems it would be best to split each business processes, physical aspects of goods as well as general information to a high level of detail. The advantage of this approach is that software packages that don't support the high level of detail can internally still manage to convert this data into the common data model. The software package simply doesn't need all the possibilities. The problem still remains for software packages that internally have such a detailed data structure that it is not possible to convert this data to the common data model. A (partial) solution of this problem is given in paragraph 7.2.2. It describes a solution for attributes that are required for one company but not for another and a solution to introduce attributes that are not part of the CDM, but are required for a company to operate.

The common data model is designed to describe information about logistic processes of transport companies. Before it is possible to give a description of the common data model, there are several terms that need to be defined. These term are used throughout the rest of the document.

Term	Definition
Transportable	Transportables are all physical materials that can be transported from one location to another. Examples are a box, an envelope or a sealed pallet.
Holder	A holder is a resource that contains one or more transportables or other holders. Examples of holders that contain transportables are cars, trailers and warehouses. If a trailer is loaded on a train then the trailer is contained in the train, which is a holder on its own.
Transholder	A transholder is a holder that also acts as a transportable.
Track	The pick-up of a transportable at a certain location and the delivery of that transportable at another location.
Remaining Track (RT)	The track from the current location to the final destination.

Table 7.1 – terms and definitions for the common data model

7.2 The Common Data Model (CDM)

The common data model can be used as a general data model to describe information about logistic processes. It has a number of primary entities which are described in this part of the chapter. This chapter would grow dramatically if all (detailed) parts of the CDM are described here. The whole description of the CDM, including examples, is given in appendix J. This chapter however does contain an overview of the major data models and their relations to give a basic idea of how the CDM is defined.

The primary goal of the CDM is to provide a model that describes the entities, attributes and relations that are common for transport companies and is based on their logistic processes for which the data is processed using a Transport Management System (TMS). The secondary goal is to design the model in such a way that it can be used for implementations of ELP to improve the automation of outsourcing and to be compatible with data models used for the development of existing TMS solutions.

The primary entities of the CDM are:

- Client: a customer of a transport company.
- Order: a request of a client to a transport company or transport broker to transport goods from one or more origins to one or more destinations.
- Transportable: physical materials that can be transported from one location to another location.
- RouteLocation: a location where a transport company has to perform a specific task.
- Holder: a resource in a transportation process that can contain one or more transportables, one or more holders or one or more transholders.
- TransportableTrack: a deviation of the pick-up, return and delivery locations of a transportable.

When the CDM is used for an implementation then a transport company is able to receive orders from clients that contain information about goods that need to be transported. The information that can be described using the CDM is based on the logistic business processes of a transport company, described in chapter 6, which means that, for example, financial aspects are absent. It would probably not be too complicated to extend the CDM with information about rates, amounts, invoices, etc.

First, the CDM is given from a non-detailed point of view to give an overview using ER diagrams. Second, parts of the overview are described in more detail by component models. Unfortunately, it is not possible to fit all detailed parts together on one page which would give a complete view. Every part of the CDM is described with more technical aspects and detail in appendix J that should be consulted if an implementation of the CDM is made.

After the more detailed description of the components of the CDM, the CDM will be viewed from an outsourcing point of view. The CDM can be used by transport companies that use outsourcing and would like to be well informed about the progress of the transport.

7.2.1 CDM Overview

The primary entities can be drawn together in an ER diagram to give the relationship between those entities. The elements that are used to draw an ER diagram are explained in detail in appendix E. The overview ER diagram has strong entities Client, Order, Transportable and Holder. Client and Holder represent long-lived business resources as they can appear in multiple orders that are mutually processed. An order can exist on its own although it should have a reference to a client because orders without a client are quite unusable. A transportable can exists on its own but it has a reference attribute constraint that it must always be contained in a holder. The final destination of a transportable can be seen as special holder that is the final holder of every transportable. The ER diagram will be split into component data models in the next paragraphs by focusing on parts of the following ER diagram.

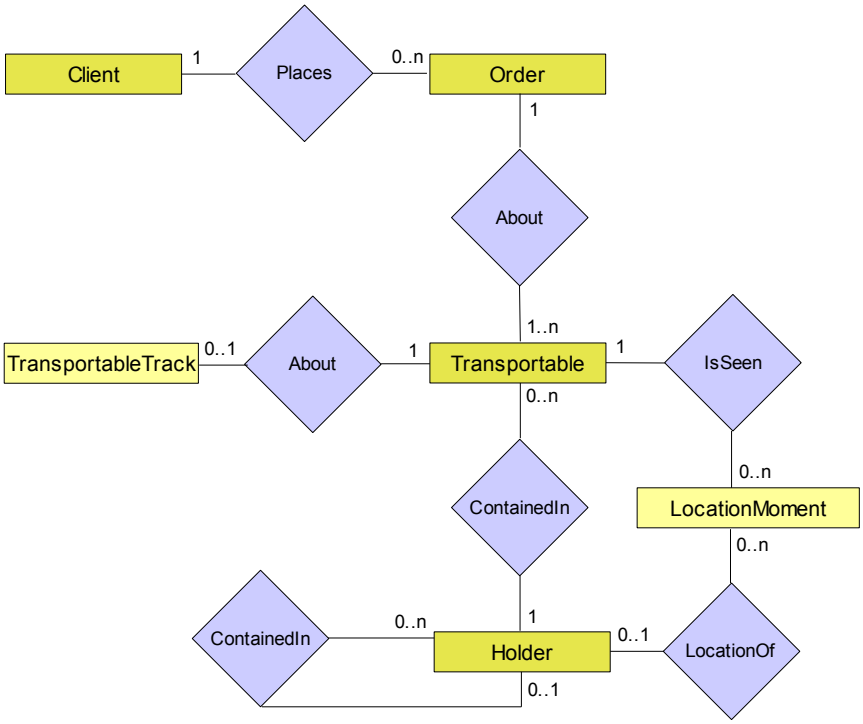


Figure 7.1 – CDM overview

An order is given by a client who is defined by the Client component model. The order should contain information about goods that need to be transported. The goods are described using the Transportable component data model such that every order should consists of at least one transportable to make the order executable. A transportable is always contained in exactly one holder, such as a truck or a warehouse. A holder, however, can contain more than one transportable. As mentioned before, a holder can be contained in another holder.

A transport company can be involved in only a part of the transport of goods from their original location to their final destination, instead of the whole transport. The Transportable component data model only contains information about the starting and ending point of the ordered transport as attributes of the Transportable. Therefore an additional component data model is needed to describe a part of the transport. The TransportableTrack items of an order are used to provide the 'from' and 'to' locations, that alter from the 'from' and 'to' of the Transportable, to describe the partial track that has to be executed by the transport company. The use of TransportableTrack is

described in more detail in paragraph 7.3.2. Every TransportableTrack must have a reference to a transportable that is part of the order.

A transportable will be stored in at least two locations during the transport process. These two locations are the original location and the final destination. It is possible that a transportable is stored or 'seen' at more than those two locations, for example in the intermediate storage warehouses used for cross-continent transports. The ER diagram has the entity LocationMoment to describe information about where a specific transportable is registered. Initially this list will be empty, but as the transports proceeds this list will be filled with locations and times where the transportable has been. All the items in the LocationMoment entity set of a transportable provide a transport history of the transportable. A LocationMoment always references to a specific transportable.

When using the CDM as a basis for an implementation then ER diagram given in figure 7.1 can be used as a basis to set-up initial database tables. The following paragraphs will take a parts of the ER diagram and describe those in more detail. Detailed information can also be found in appendix J.

7.2.1.1 The Client data component model

Every transport company has clients that place orders at the transport company. The CDM consists of a component model to describe client information. A client can be a natural person or a company. Because a natural person can be seen as a contact person of a company, the client of an order is based on a data model that describes a company. Another possibility is to introduce an ISA superclass for Client with Person and Company as subclasses. This alternative is not chosen because, from an Object Oriented point of view, Company is inherited from Person so using the Company data structure provides all aspects of the Person data structure. The detailed ER diagram of Client is displayed below.

☞ *functional requirement RQFuncInf4*

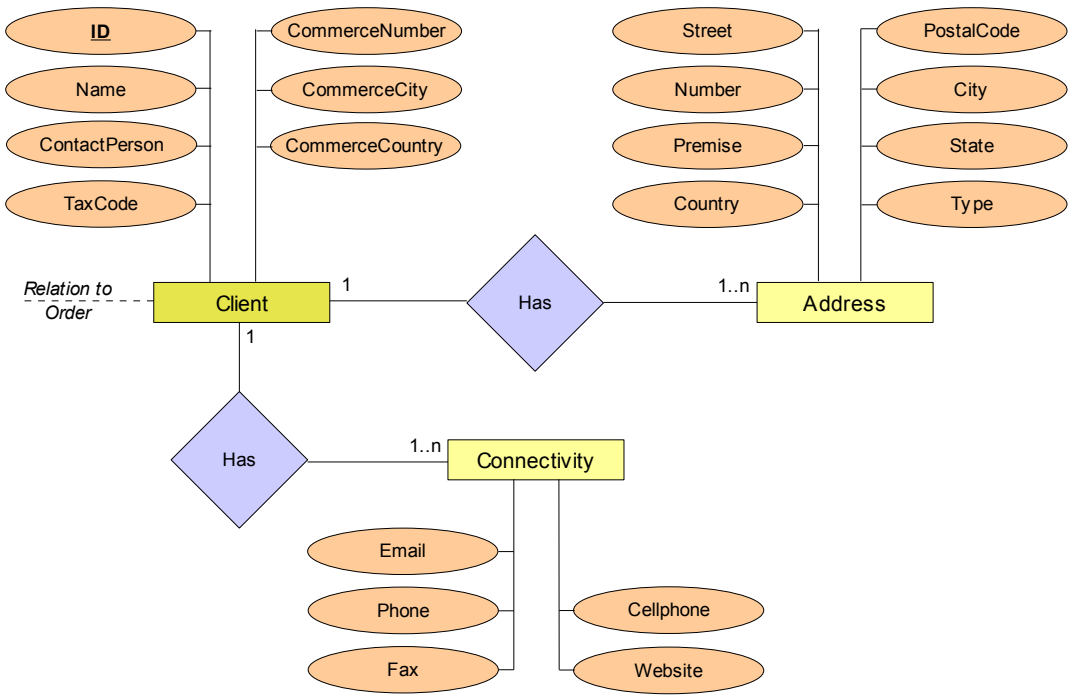


Figure 7.2 – client entity relationship

The attributes in the ER diagram are common to describe information about companies. They are compatible with Microsoft Outlook and therefore usable with Microsoft Exchange to provide easy integration. A remark can be made to the relationships between Client and Address. One client can have multiple addresses where the Type attribute indicates whether it is a settlement address or an invoice address. A client has at least a settlement address; the invoice address is optional if it differs from the settlement address.

7.2.1.2 The Order data component model

A client, that can also be another transport company, has to be able to place orders which are described by the Order data component model. The detailed ER diagram of an order is displayed below.

The order is uniquely identified by the identifier given by the ID attribute and should have a reference to a client that has placed the order at a certain moment in time (Moment). The order has a list of transportables that describe all the goods that need to be transported. The data component model of a transportable will be described later. The ReferencePerson attribute can contain the name of the person that can be contacted at the clients if questions or problems arise.

☞ *functional requirement RQFuncInf2*

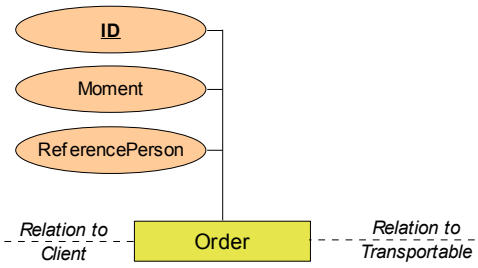


Figure 7.3 – order entity

7.2.1.3 The Transportable, TransportableTrack and RouteLocation data component models

The data component model of a transportable describes all kind of information about goods. For example, this information contains the sender, the destination and specific properties that are used to decide what resource to use such as weight and dangerous good numbers. The Transportable data component model has many fields from which now only the most important are displayed. The following figure illustrates the component model:

☞ *functional requirements RQFuncInf3, RQFuncInf7 and RQFuncTra4*

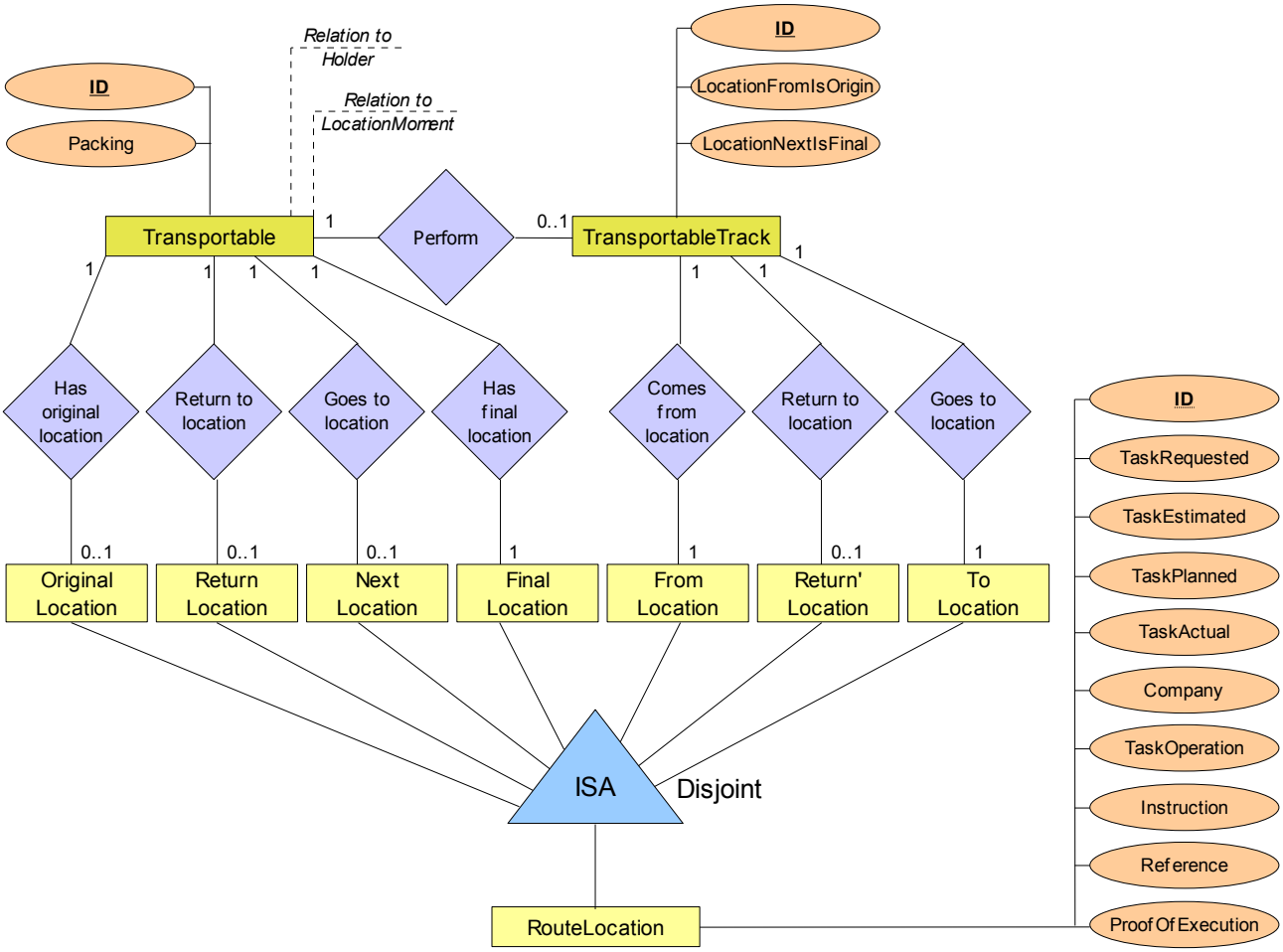


Figure 7.4 – transportable and transportable entity relationship

Every transportable is identified by a unique identifier. The packing attribute describes the packing where the goods are packed in, for example a box or an envelope. Transportable and TransportableTrack entities both have references to RouteLocation entities. For a transportable they describe the location where the goods are originally coming from, need to be returned to in case they are undeliverable, are currently going to and finally need to be delivered. There is a constraint on the 'Original Location' and 'Return Location' entities that is not part of the figure. The constraint is defined as: if a transportable hasn't got a 'Return Location' then it must have a 'Original Location'. The reason for this constraint is that there has to be at least one location to return the goods to in case they are undeliverable. Due to competition, the original location is often hidden to preserve client information.

It is common that the transport of goods is done by multiple transport companies or employees. The TransportableTrack entity contains information that is specific for a transportable for only one transport company or employee. The TransportableTrack component model describes the track that has to be executed by a specific company or employee. In short, TransportableTrack is introduced to deviate a transport from the locations described by a transportable. The TransportableTrack items can be seen as the parts in which an order is split in the business process described in paragraph 6.3.2.

Information about the pick-up and the delivery of goods is described using the RouteLocation data component model. It describes a task that has to be performed at a certain moment at a certain location. Typical information that is described, is an address, the kind of operation such as loading or unloading and information about the preferred time this has to take place. The TaskRequested, TaskEstimated, TaskPlanned and TaskActual attributes are TimeWindow data component models that, respectively, give the time window in which the task is requested to be performed by the client, probably performed, planned by the transport company and actually performed. The ProofOfExecution attribute is a data component model that contains information about the person who signed if the task is completed.

Every RouteLocation entity has specific information about the task that has to be performed at a specific location. Since all the locations of a Transportable and TransportableTrack have different meanings, a RouteLocation can only apply to one of these locations. This is indicated by the disjoint ISA relation.

7.2.1.4 The Holder and LocationMoment data component model

The data component model of a holder describes information about all kind of resources that are used during a transport process that can contain a transportable. Typical examples of holders are vehicles and warehouses, which can hold goods. The data model of holder is illustrated in the following ER diagram:

☞ *functional requirements RQFuncTra1 and RQFuncTra3*

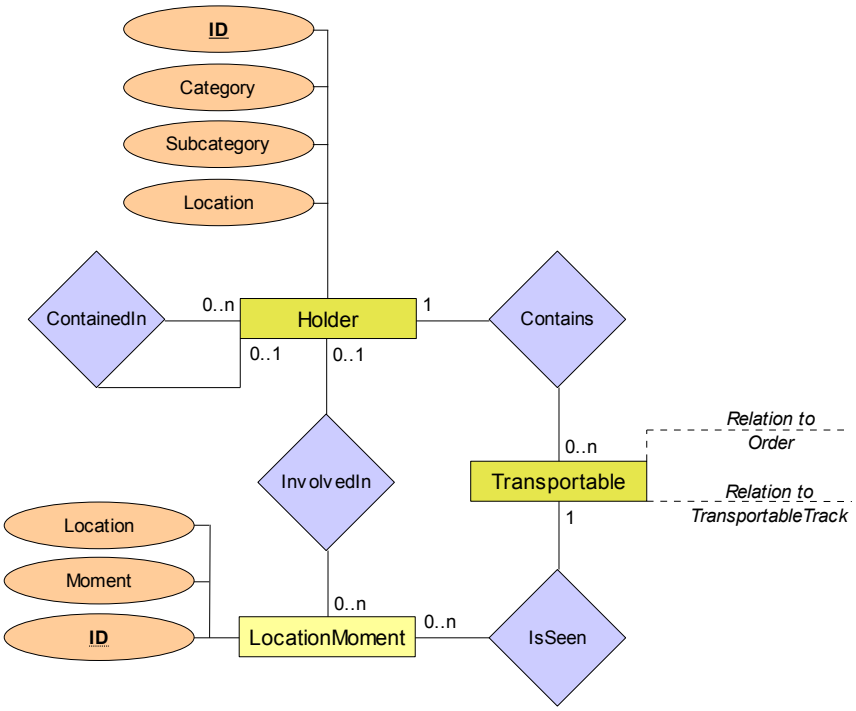


Figure 7.5 – holder and transportable entity relation

Every holder is uniquely identified by an ID. A holder is part of a category and a subcategory, described the appendix J, and describes what kind of holder it is, such as subcategory 'Warehouse' in the category 'Buildings'. The location of a holder gives information about the place on earth, such as the GPS location of a vehicle or an address of a building. It is possible that a holder is contained in another holder. When a holder is contained in another holder then the ContainedIn reference gives the identifier of the holder where the holder is contained in, for example a trailer that is contained in a train.

The history of a transportable, such as the locations where a transportable has been stored during the transport, is described by the entity LocationMoment. This entity describes the moment that a transportable was at a specific location. It is possible to refer to a holder to provide additional information that is part of a holder but not of a LocationMoment.

7.2.2 Extensions to the CDM

The common data model provides, as the name already suggests, common properties of a data model that can be used for the development of a TMS. It cannot be proven that the model is suited for all transport companies and gives a reason to think about changes that can be made to the CDM without interfering with its basic definitions. This is exactly where the first word of ELP, Extendible, comes into place. There consist two problems that are solved in this paragraph using extendability.

When an order is placed at a transport company then the following problems can occur:

- A client doesn't know which attributes are required by a transport company: an attribute is required by a company to execute the order but the value of the attribute is not given. The required attributes don't need to be the same for every company.
- A transport company requires specific information that is not part of the order: the CDM doesn't provide the attributes that describe the specific information.

7.2.2.1 Required attributes

Together with the first problem one can ask what information is required. Although the descriptions in appendix J provide information about required attributes, it would be preferable if a transport company can define the required attributes on its own. A solution to this problem can be to introduce an extra layer on top of the CDM that defines which attributes are required (must-have), recommended (should-have) or optional (could-have). This extra layer overrules the CDM except for primary or foreign key attributes.

First it is assumed that the CDM has a version so that future changes can be made and detected within ELP. When assumed that the described CDM has version 1.0.0 then the “requirement layer” should have an reference to CDM version 1.0.0. The requirement layer itself must also have a version for the same reasons as the CDM. A difference between the CDM and the requirement layer is that the CDM cannot be altered by a transport company, but the requirement layer can. It is possible that the company uses a different requirement layer for different customers. Therefore the requirement layer must have an identifier that is given by the transport company and is globally unique to protect it being confused with other requirement layer provided by other companies with the same name. Chapter 10 describes unique identifiers with ELP (ELP-Id's). A requirement layer can be identified by the following set of key-value pairs:

{Version=1.0.3,CDMVersion=1.0.0,ID=Foo-bar}

The content of the requirement layer data can contain information about the attribute

'ContactPerson' of entity 'Order' being required. This paragraph doesn't go into detail about how to describe these required properties.

7.2.2.2 Extending attributes

The second problem involves attributes that are not part of the CDM, but are required by a transport company. Examples of attributes that can be required are:

- An order registration number that is obliged by the government in a country
- Information required for customs documents

These examples are not a coincidence, because they are examples of attributes that can be required for business processes that are defined to be out of scope of ELP for now, see paragraph 3.2. Although being out of scope, if it is possible to extend ELP with these attributes then it shows a clear example of the power of ELP specifications being extended and therefore more suitable for custom business processes.

Q204

The solution to provide extendibility to the CDM can be analogue to the solution for defining required attributes, namely by introducing an additional layer that describes additional attributes. This layer is called the “extension layer” and exists on top of the CDM, but below the requirements layer. The reason for this is that, if the extension layer is on top of the requirement layer, it is not possible to provide any information about an additional attribute being required. An addition of attributes and requirements is illustrated in figure 7.6 below:

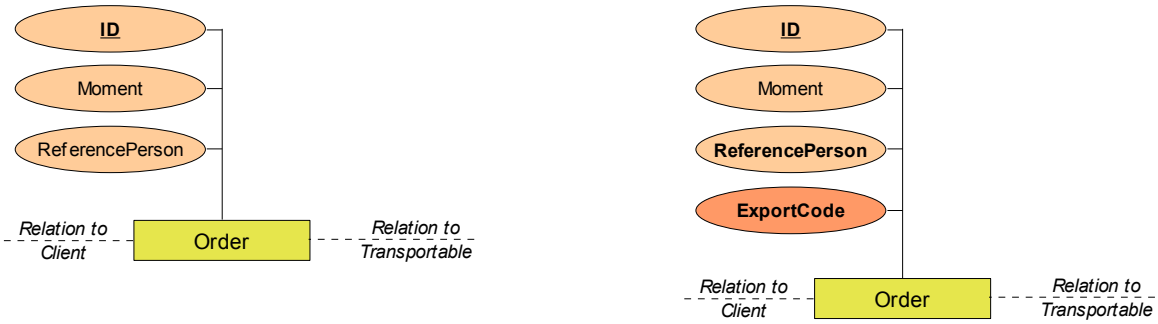


Figure 7.6 – order entity extended with an extension layer

First, the extension layer is applied to the CDM. In the illustrating example above the extension consist of the attribute 'ExportCode' that, for distinction reasons, is drawn using a darker orange color. Second, the requirement layer is applied to the current model. In this example the layer defines that 'ReferencePerson' and 'ExportCode' are required (must-have) attributes, illustrated by bold names. Should-have and could-have attributes can, respectively, be written using a italic and normal font.

When a transport company needs additional attributes or values for required attributes then clients of this company need to know about it. One way of providing these requirements is by adding them to ELPNS. Another way is to set-up a method to request them from a transport company. A complete solution is not part of this document.

7.2.2.3 Future extensions

The previous two paragraphs described two kind of extension that can be used to make ELP wider applicable. It is not excluded that more extensions are needed to let ELP work for every company.

Other extensions that can be thought of are the introduction of new entities to the CDM or even a complete new data model that can be used to share information about the locations of a companies fleet. ELP can provide the framework to describe the data models that can be used together with shared entities.

7.3 CDM usage and outsourcing

This chapter so far focused on the use of the CDM by a single company to describe the data model that it could use. This paragraphs focuses on on the use of the CDM by more than one company and especially for using it with outsourcing. The outsourcing of transport between brokers and transport companies is one of main aspects of ELP. The outsourcing of orders has influenced the focus on the design of the CDM that is also described by the second goal of the CDM.

7.3.1 Outsourcing and the ELP Identifier

Outsourcing involves the exchange of data between two or more transport companies. Information that is sent from one to another. Examples of this information can be quotations, orders, reservations and progress updates. When an order is outsourced then the outsourcing company has to store information about the order and the transport company or companies it is outsourced to. The track that is outsourced can be described by the TransportableTrack entity, but this entity doesn't provide any information about the transport company that is executing that part. Also, the outsourcing transport company must have placed an order at the other transport company, which is also not described by the CDM so far. The CDM needs to be extended to describe a model that can also be used for outsourcing.

Q003

Many parts of the CDM can be used to extend the model for outsourcing. First, the company where an order is outsourced to can be modeled by the Company component model. Second, a new entity 'Outsourced order' should be introduced to model orders that are outsourced. The content of the order is described by the Transportable and TransportableTrack entities of the CDM. The outsourced order is about one or more transportables that are transported by another company over the tracks that are described by the transportabletrack of each transportable.

When a transport company would like to outsource an order then it needs a way to identify the transport company where the order is outsourced to. Also, it needs information about how to contact this company. A solution to this problem is the ELP Identifier (ELP-Id). The ELP-Id is a unique identifier for each transport company. The information that is needed to communicate and exchange data with this company and can be offered by a protocol analogue to the existing DNS protocol. Instead of resolving a domain name to an IP address this protocol can provide a lookup from an ELP-Id to the (technical) information needed to communicate with a company. Instead of Domain Name Service this lookup service is now referred to as ELPNS. The assigning of ELP-Id's and providing the ELPNS is not part of this document, but is trivially a centralized administration.

The extension to the CDM as well as the ELP-Id create the following model:

☞ *functional requirements RQFuncInf1 and RQFuncInf4*

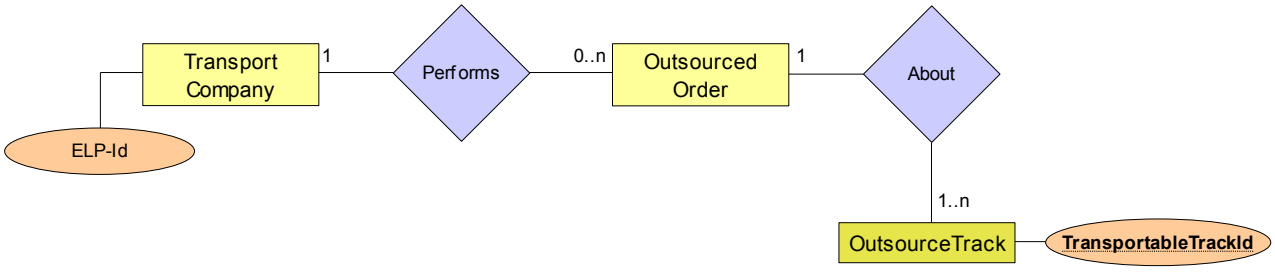


Figure 7.7 – CDM extensions

Figure 7.7 displays the strong entity Transport Company. This entity is equal to that of Company except that it has an extra attribute ELP-Id. The content of this attribute is needed to communicate with this company. The Transport Company can perform zero or more orders that are outsourced to this company. An Outsourced Order is about one or more Outsource Tracks that are referred to with the TransportableTrackId attribute. This attribute is a foreign key to a TransportableTrack. Because every TransportableTrack has a reference to a transportable, the outsourced order also has references to transportables using the TransportableTrackId of its OutsourceTracks. The relation between OutsourceTrack and Transportable track can also be given using a one-to-one relation between these entities.

It is possible that an order is outsourced to more than one company. All companies perform a part of the complete transport. In this case there are more 'Outsourced Order' items of which OutsourceTrack refers to different TransportableTrack items, but these TransportableTrack items refer to the same transportable.

When an order is outsourced then the outsourcing company sends the information about the transportable as well as the transportabletrack to the executing company. The transportabletrack belongs to the outsourced order and refers to the transportable. The company where the order is outsourced to now has the information about the goods as well as the from/to information that is needed for the execution.

The outsourcing company would like to receive updates about the progress of the execution. The executing company has to know how to contact its client thus the Company component model should be extended with an ELP-Id of the client. This means that the Transport Company and Company entities are equal.

During the execution of the order by another transport company it can add items to the LocationMoment entity set as well as change the holder of a transportable. These changes form updates on the progress of the order and the outsourcing company would also like to receive them. A solution to distribute the updates on the progress to all the involved participants is to share the data of a transportable, its LocationMoment items and the current Holder. This combination is now referred to as 'shared transportable'. When one participant changes the shared transportable then the shared transportable data is also updated for the other participants. Using shared transportables, it is also possible to outsource an order multiple times where only the number of participants sharing the transportable grows.

It is possible that a transport companies requires values for attributes of the CDM to be given or values for attributes that are not part of the CDM. The next paragraph focuses on required attributes as well as additional attributes that are needed by transport companies to execute an order.

7.3.2 Multiple outsourcing

As an alternative of outsourcing to a single company, it is possible that the transport is outsourced to multiple transport companies. In this case, equal to use case 3, Company A decides to place an order at Company B to transport the goods from the client its warehouse. Company A also places an order at Company C to transport the goods from the warehouse of company B to the final location. Since the data of transportables is shared among the transport companies, the pick-up and delivery locations of the transportables are not correct for every transport company. In case of multiple outsourcing, company A includes instances of the TransportableTrack data model to the orders placed at company B and company C. These instances describe the part of the transport that has to be done by a specific transport company. This implies that they are different for company B and C, but are both known by company A. The following figure illustrates multiple outsourcing to describe the data components used, locations and the holders of the transportables.

☞ *functional requirements RQFuncInf7 and RQFuncTra3*

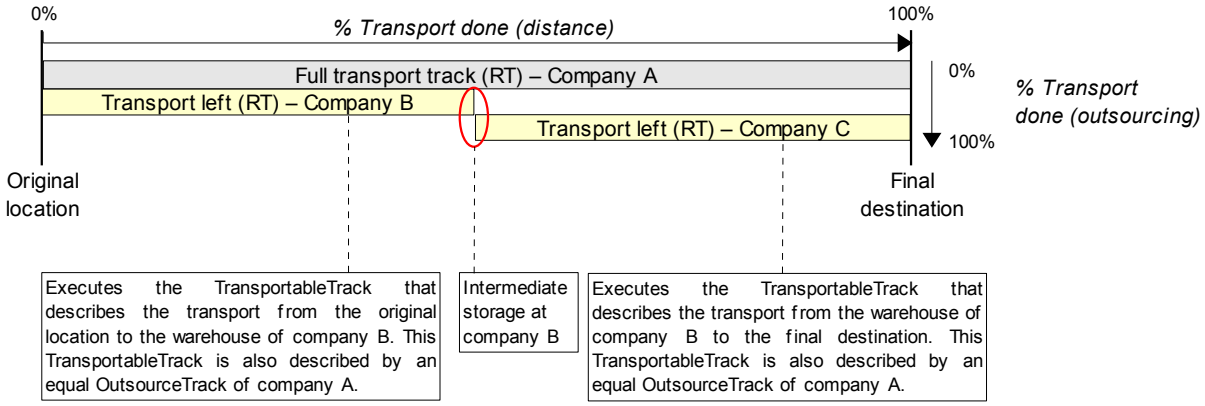


Figure 7.8 – overview of transport executed by multiple companies

Data components of participants

Company A

- Order: transport goods from the original location (OL) to the final destination (FD) for client C
- Transportable: described the goods and their OL and FD
- OutsourceTrack B: describes the part of the transport outsourced to company B
- OutsourceTrack C: described the part of the transport outsourced to company C

Company B

- Order: transport goods from the OL to the intermediate storage for company A
- Transportable: described the goods and their OL and FD
- TransportableTrack: describes the transport company B has to execute

Company C

- Analogue to company C

Locations and holders

The transportable has, assuming that different transportation means are used, five different holders and three different locations during the transport track . The five holders and locations are:

1. The original location (1st location)
2. The transportation means between the OL and the intermediate
3. The intermediate storage (2nd location)
4. The transportation means between the intermediate storage and the FL
5. The final location (3rd location)

7.4 TransHolder: the holder that can be transported

7.4.1 Transholder functionality

The previous paragraphs described how outsourcing takes place when a transportable is transported by one or more transport companies. Although this would be sufficient for general outsourcing purposes, described in use case 1 to 4, it is not sufficient for use case 5. This is the first use case where a transholder is introduced. A transholder can be seen as a long-lived resource that is used to combine multiple transportables into one transportable. So, the transportables are contained in the transholder. An example of a transholder, that is well-known, is a pallet where boxes are stacked upon. The two main differences between a holder and a transholder are the fact that the transholder is transported including an origin and a final location

and that a holder doesn't contain other transportables.

The transportables that are contained in a transholder can have different final locations and usually have a partial track in common. The transport company that created the transholder, i.e. put the boxes on the pallet, knows which transportables are contained in that transholder. After the creation, the transholder can be seen as a transportable that is described by the Transportable data model of the CDM.

As a transholder is in fact a new transportable, there arises a problem for the track and trace functionality. Transport companies that only transport the transholder do not know about transportables that are contained in the transholder. This means that these companies also don't share the data of the transportables and thus can not update, for example, the holder and the LocationMoment items. There are several solutions for this problem:

1. Ignore the problem and accept the missing track and trace information.
2. Provide the track and trace information about the transholder to the companies involved in the transport of the transportables that are contained in the transholder.
3. Assume that the track and trace information about the transholder is an addition to that of the transportables.

The first solution is not acceptable since one of the primary aspects of ELP is to provide good track and trace functionality, see paragraph 4.3. The second solution can be acceptable and feasible, but doesn't take *separation of concerns* in consideration. There is no need for the client to know that one or more of its transportables are contained in a transholder. The third solution doesn't work, because the companies that are involved in the transport of the transholder do not know about the transportables. However, it is possible to create a feasible solution by combining solution two and three. The key role in this combination is performed by the transport company that created the transholder. This company has up-to-date information about the transholder as well as the transportables. The solution would be that if changes are made to the transholder then this company makes the same changes to the transportables. For example, when the holder of the transholder changes to warehouse A then the holder of all the transportables also changes to warehouse A. Appendix F consists of a set of rules for the administration of holders, transholders and transportables that have to be obliged by transport companies to guarantee up-to-date track and trace information.

The following figure illustrates a transport process in which a transholder is involved. The actual transport can be extracted from the cyan boxes from top to bottom. The cyan boxes on the left describe the transport of the transportable when it is not part of a transholder and these on the right describe the transport when it is. The red boxes illustrate the transitions of containment of the transportable. The progress information that company A sends to the customer is left out for simplicity, but the aspect that it has this information available can be derived from the blue rhombuses that represent the exchange of information about the 'shared transportables'.

☞ *functional requirements RQFuncInf8 and RQFuncTra2*

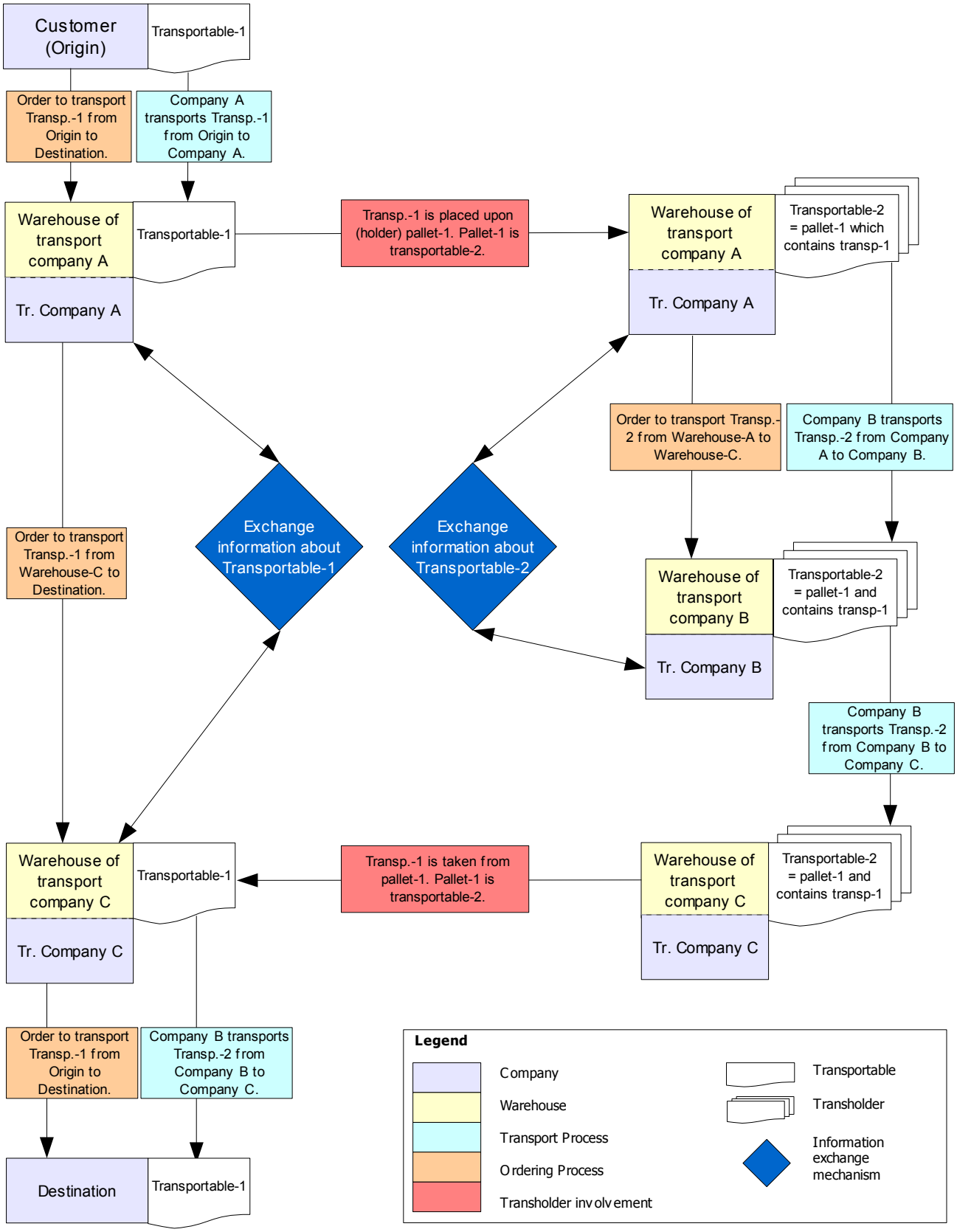


Figure 7.9 – outsourcing using transholders

7.5 Summary

The Common Data Model, abbreviated as CDM, provides a model that can be used to describe clients and orders in such a way that existing information systems are able to map their existing proprietary model to the CDM. The CDM introduces the entities Transportable and Holder to model goods, including their specifications and addressing, and resources that are currently contain them such as warehouses and vehicles. As an extension to these entities, the TransHolder entity can be used to model the situation where goods are contained in other long-lived resources such as sea containers. Due to required or addition proprietary attributes by existing information systems, a way to extend the CDM is given. Finally, the CDM is extended with entities to support situations where multiple transportation companies only execute a part of the actual transport or outsource such a part to another transport company. The addition CDM entities TransportableTrack, TransportCompany, OutsourcedOrder and OutsourceTrack provide support for describing information that is introduced when an order is outsourced. Outsourcing introduces the exchange of information to provide accurate information what can be described using an information exchange overview with rules that have be followed. To be able to keep track of the goods and to exchange this information, the entity LocationMoment is added to the CDM that describes the location of goods during the execution of the order.

8 Communication between companies

8.1 Introduction to communication

Q501
Q502

The previous chapters described the updates of the shared data of a transportable by two or more companies. To provide any update there has to be a way to communicate between those companies. Paragraph 4.2 indicated that private communication lines cannot be excluded. It speaks for itself that the Internet is currently the most used means of communication between electronic devices such as personal computers. It can therefore easily be assumed that the Internet is used for communication between companies that share data of a transportable. However, this ignores the existence of private networks that larger international transport companies can have and it cannot be excluded that companies exist that have the ability for digital communication, without using the Internet. Paragraph 4.2 also mentioned different transport method when the Internet is actually used what indicates that the assumption of using the Internet is not precise enough. Because of these reasons, this chapter focuses on the creation of communication networks that can easily be represented by the Internet, but don't take the everybody-can-talk-to-everybody property, always-connected property and a predefined transport method for granted. Instead, the actual means of communication is left out of scope and a communication line between two companies indicates that, using an arbitrary means of communication and transport method, it is possible to send each other messages (full duplex).

In this chapter are companies replaced by nodes in a network that have to be able to communicate. When two companies are able to communicate directly then they are connected with an edge. The communication network between the nodes can be like the order scheme. This results in a tree (or acyclic graph) with nodes connected with communication channels. The communication networks of use case 3 and 4 are illustrated below.

Figures 8.1.a-b - on the left (8.1.a): use case 3; on the right (8.1.b): use case 4

The communication channels provide a way to send messages from one node to another. A communication channel must always be bidirectional to enable the nodes to 'talk' to each other and therefore the edges are undirected. A communication network with a topology comparable to that of the order scheme guarantees that a path, and therefore a (indirect) communication channel, exists from one node to another. However, the topology of the communication network doesn't necessarily need to be equal to the order scheme. The first part of this chapter focuses on several alternatives of creating a communication network.

The outsourcing of an order doesn't only consist of placing an order at another transport company. It also consists of sending progress information to the transport company that outsourced the order described by the “execute order” sub-process in paragraph 6.3.3. The previous chapter introduced information updates about data that is required by the nodes. One of the purposes of the updates is that a transport company is able to react on events that can change its part of the transport. The second part of this chapter focuses on updating data at other nodes within the communication network.

Reasons for communication

The introduction described a communication network between nodes that belong to an outsourcing scenario. The dotted arrows between the two pools described in paragraph 6.1 provide three reasons for communication, namely:

1. Requesting and providing quotations
2. Placing orders by accepting quotations
3. Sending and receiving progress information

It has to be pointed out that these reasons also apply to communication between a client and a transport company and not just for the situation where one transport company outsources an order to another. The first two reasons for communication imply that communication is needed by at least two participants, namely the node that places (or outsources) the order, i.e. the parent, and the node that accepts the order, i.e. the child. This applies to use case 1 and 2. However, when more participants are involved, the question rises which nodes should be able to communicate with each other. There are two situations where more than one node is accepting an (partially) outsourced order.

The reasons for communication deliberately don't include the creation, changing and cancellation of reservations and orders, because these belong to functional requirements RQFuncBus4 and RQFuncBus6 that have a low priority. The reasons for communication can be extended if these requirements are considered by research or development in the future.

Multiple outsourcing with a single level order scheme

The first situation consists of a transport company that outsources an order to two or more companies and where the order scheme has a single level of outsourcing; this applies to use case 3. The first two reasons for communication define that a communication channel has to exist between the order placing and accepting node. When assumed that direct communication channels are possible then the topology of the network is equal to the order scheme except that the arrows represent bidirectional communication channels. The third reason for communication is also accomplished by this topology, because when every node is sending progress messages to its parent node then every node has the information needed to be accountable. This corresponds to the second goal of the use cases, see paragraph 5.1.

The topology equal to the order scheme implies that company B and C of use case 3 do not communicate or at least not directly. Since these two companies do not do any business together this should not be a problem for communication reason 1 and 2. Reason 3 however introduces a practical reason why company C would like to communicate with company B. This reason is an event during the transportation process of company B that has an effect on the transportation process, such as the transportation- and resource schedules, of company C. This implies that company C likes to receive the progress messages of company B, but not vice-versa, and therefore (in)direct communication between them can prevent transportation failures.

Multiple outsourcing with a multilevel order scheme

The second situation consists of more than one transport company that outsources (a part of) its orders; this applies to use case 4 and 5. The communication requirements for placing and accepting orders are equal to those described in the previous paragraphs except that more communication channels exist on multiple levels. However, a difference can exist in the need to receive progress messages. For example, when a delay occurs at the transportation process of company D then this doesn't necessarily imply that the transport schedule of company C changes but it can also not be denied. The practical reason for communication between company B and C from use case 3 suffers the same insecureness although on a smaller scale.

In general, independent whether an event influences some schedule or not, it can be concluded that the progress messages have more functionality then just providing information to a parent node to be held accountable. The progress messages also provide useful information for any downstream transport company presented in the flow of goods schemes of the use cases.

8.2 Communication requirements

The reasons for communication in the previous paragraph describe requirements of the communication network between nodes. These and additional aspects can be summarized in a list of requirements of the communication network.

Communication channels between order placing/accepting nodes
To place an order there has to be a communication channel between the node, that can also be a client, that is outsourcing (a part of) an order and the node that could accept the order. This communication channel is also needed to request and provide quotations as well as send progress messages due to responsibility. [RQP001]

Knowledge about downstream nodes
Progress message provide useful information for downstream nodes in the flow of goods scheme. Nodes need to know what nodes downstream in this scheme to be able to send progress messages to them. [RQP002]

Communication channels between succeeding nodes
The previous paragraph described the need for progress messages to be received by downstream nodes within the flow of goods scheme of the use cases. To be able to receive these progress messages there has to be a (indirect) communication channel between the sender of the progress message and the transport companies that are downstream in the flow of goods scheme. [RQP003]

Multiple communication channels per node
A transport company, represented by a node, has to be able to communicate with more than one node in parallel. This is needed because it can request quotations from many nodes in parallel as described by the business process in paragraph 6.2.1. [RQP004]

Real-time addition and removal of nodes
A transport company outsourcing an order increases the number of companies within the order scheme. Requirement RQP003 introduced the need for a communication channel between a new company and those earlier in the flow of goods scheme when it is added at the end of the flow of goods scheme. The added company has to receive the the progress messages sent by a company earlier in the flow of goods scheme and thus there has to be an administration consisting of the nodes that require to receive progress messages. [RQP005]

Support for nodes being unavailable
Due to, for example, communication problems, it is possible that a node is unavailable for some time. The network of nodes can consist of mobile nodes that are not able to communicate for some time. However, it is taken for granted that all nodes are available 50% of the time. [RQP006]

◈ Q301 Q503

8.3 Communication network topologies

To describe several alternatives for the communication network topology, use case 4 is used as a reference. This use case is chosen because it contains multiple companies outsourcing orders on multiple levels. Topologies that suit this use case also suit less complex ones since these are only simplifications of it.

The order schemes of use cases can be drawn using trees where nodes represent companies and edges represent communication lines. To simplify RQP003 to this tree, it is assumed that all the nodes are presented in an order from left to right and that the nodes drawn using a bold circle form the actual transport process described by the flow of goods scheme. This means that in use case 4 the first company actually executing a transport is company D followed by E and C (all drawn using bold circles). Also, it is possible that transportation takes place in parallel using multiple vehicles or multiple outsourcing. An example of the latter is such a big amount of boxes that the order is partially outsourced to one transport company and partially to another. It is assumed that, if this transportation track is the final transportation track, it is accepted by the receiver to receive the goods by multiple deliveries. Parallel outsourcing is drawn in the trees by two bold nodes that appear on the same vertical position. The following figure illustrates the tree of use case 4 with communication lines and it is extended with company F to include parallelism.

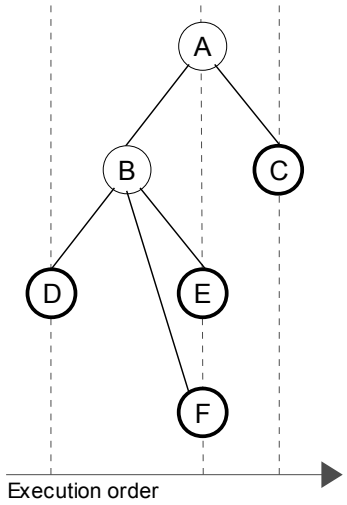


Figure 8.2 – parallel transportation

Chapter 7 described that every transportable of an order has its own transportation track and therefore it is possible for every transportable to be transported from its original location to its final destination using different transport companies and/or vehicles and/or routes; even in parallel. It is not excluded that one transport company appears more than once when it accepted multiple outsourced orders that all represent a part of the complete transport.

Requirements RQP001 and RQP003 describe communication channels between:

- Nodes that have an order placement relation
- Nodes that send progress messages that need to be received by downstream nodes within the flow of goods scheme

Using RQP001 and RQP003 several alternatives can be used for the communication network. These alternatives are distinguished by direct/indirect communication channels and the introduction of additional nodes that only function as message pass-through. An indirect communication channel is defined as a communication channel from one node to another where one or more nodes, that pass through messages, exist between them. It has to be pointed that when nodes are mentioned, these exist at the application layer of the OSI reference model [Tanenbaum]. This implies that when two nodes have a direct communication channel it doesn't mean that the communication channel is also direct on the transport layer.

Several resources, such as [TanenbaumProxy], [WikiProxy], [WikiGateway] and [PCMagGateway] do not fully agree on the definitions of proxy, gateway and router. Therefore the following specific definitions are used:

Term	Definition
Proxy-node	A node that is not the sender nor intended receiver of a message and passes through messages it received, either with or without routing functionality (see Router).
Router	A proxy-node with three or more communication channels and some kind of intelligence to decide to which communication channel it will forward a received message.

The communication network, where the client is absent for simplicity, always starts with a single node that is the root of the tree. When (partial) orders are outsources then new nodes are added to the network, according to RQP005. The topology of the communication network is determined by the building steps that describe what communication channels are added when an additional node is added to the network. These building steps are redone every time a node is added. If a communication channel between two nodes already exists then this remains the only communication channel.

The following elements are used for a graphical representation of a topology:

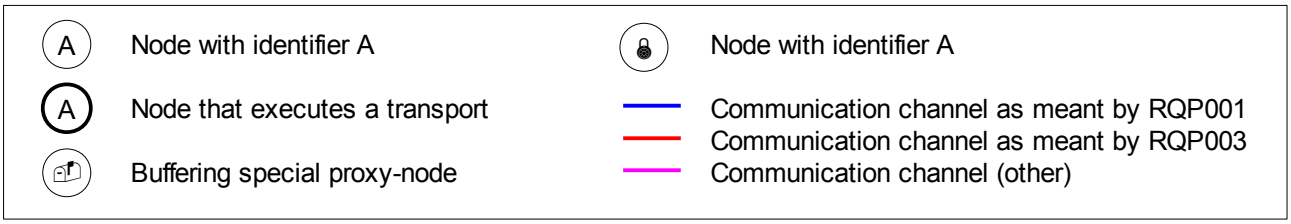


Figure 8.3 – elements used within topology illustrations

Network topology approach 1: direct communication channels

The primary aspect of using direct communication channels is that all messages sent from one node to another do not pass any other node. This paragraph describes two methods to acquire a network topology using only direct communication channels. It is emphasized that these topologies represent a network topology where the execution of transport can be derived from as described by the previous paragraphs.

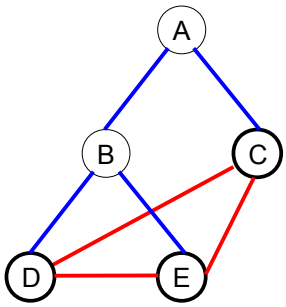


Figure 8.4 – topology 1a

Building steps for topology 1a:

1. Introduce a communication channel between every order placing/accepting pair of nodes [RQFuncInf1]
2. Initiate a communication channel between the new bold node and other bold nodes by a mechanism where the new bold node 'broadcasts' its appearance.
3. Every bold node that does not have a communication channel with the new bold node creates one.

Building step 2 is in fact indirect communication between nodes, but is not considered as indirect communication because the broadcast messages are used to build the network topology and not for one or more of the the three reasons for communication.

An important aspect of this topology as well as some of the downstream (indirect communication) topologies is business logic. Within the order tree, that is equal to figure 8.4 without the red edges, the edges also represent a nodes responsibility. This means that a node from a business, i.e. non-technology, point of view can only be held accountable by its parent. The direct communication channels however introduce an additional task for a node that it is assumed to perform. This additional task is to send progress messages to nodes it has no business relation with. For example, node D is assumed to send progress messages to node C as addition to those to node B. The progress messages to node D are related to RQP003 while the progress messages to node B are related to RQP001. The latter are the only messages that are required from a business point of view. In short, within network topology 1a there is no supporting business obligation for node D to send progress messages to node C.

Q004

From the previous paragraph two questions can be asked, namely:

- A node is not obliged to send progress messages to nodes that it doesn't do business with according to the order tree. Is there a way to introduce this obligation?
- Is an obligation to send progress messages to non-business participant acceptable in practice?

The first question can be answered positively, namely by introducing an additional part to the legal agreement between an outsourcing the order accepting participant. This part includes the obligation that every node has to forward broadcast messages as well as creating communication channels if needed. However, the answer to the second question can make the answer to the first question worthless. This question cannot be answered at this point although it can be assumed that any technology that supports (current) business processes will be accepted more easily. This implies that a technology that supports the obligations according to the order tree is an advantage. If a network topology has this advantage then it is entitled as ' it follows the order scheme'.

Q107

Advantages of topology 1a:

- No routing mechanism for progress messages required
- No proxy-node(s) that can fail
- The distance between two nodes is $O(1)$

Disadvantage of this topology:

- The red communication channels already exist indirectly. Additional communication channels require more resources.
- It has to be possible to create direct communication channels between all nodes.
- Bold nodes have to keep an administration of nodes they have to send progress messages to. This administration has to be updated when a bold node is added or removed.
- (Confidential) business process information is published. For example, node A might not like that node D has knowledge about node C as part of the transportation process due to competition.
- Coordination of the creation of new communication channels is complex because many nodes, including ones that do not become a part of the channel, are involved. For example,

A has to send a message to B that it added C. Now B has to send a message to its children that C is added and that, if one of them is a leaf, it has to establish a communication link to C, et cetera. This is an example of a possible broadcast mechanism.

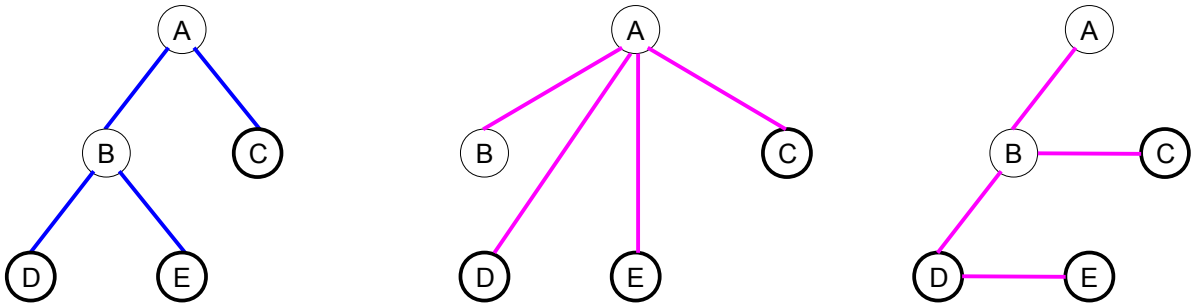
The building steps of topology 1a can be rewritten to one more general rule (1b), namely:

1. Introduce a communication channel between every new node and all existing nodes, regardless whether a node is bold or not. This is also done using a broadcast mechanism.

The result of network topology 1b is a complete graph with many unused communication channels, because they are not needed according to RQP001 and RQP003. The number of communication channels in this network topology grows quadratically ($O(n^2)$), namely n(n-1)/2 where n is the number of nodes.

Network topology approach 2: indirect communication channels

The primary aspects of indirect communication channels is that, as opposite to direct communication channels, messages, as meant by the reasons for communication, can pass other nodes that are not the sender or intended receiver. If an indirect communication channel between two nodes already exists then there is no need to introduce an additional communication channel. The following figure illustrates several approaches to create a network topology for indirect communication.



Figures 8.5.a-c – topology 2a (8.5.a), topology 2b (8.5.b) and topology 2c (8.5.c)

Building steps for topology 2a:

1. Introduce a communication channel between every order placing/accepting pair of nodes [RQP001]

Building steps for topology 2b:

1. Introduce a single communication communication channel when a node is added to the order tree. This communication channel is created in the network topology between the node added to the order tree and the root of the network topology. If the outsourcing node is the root of the order tree, then introduce a communication channel between the root of the network topology and the added node.

Building steps for topology 2c:

1. Introduce a single communication communication channel when a node is added to the order tree as follows. If node A has no children in the order tree and it outsources (a part of) its order to node B then this communication channel is created in the network topology between A and B. Node A marks B as its *network topology extension node*.
2. If node A already has children in the order tree and it outsources another part of its order to node C then a communication channel is created in the network topology between node C and the *network topology extension node*. Now node C is marked as *network topology extension node* instead of node B.

The primary aspects of network topology 2a and 2b is that 2a results in a topology equal to the order tree while 2b results in a star topology with the root of the order tree as central node. Topology 2c has as primary aspect that every node has a maximum of three communication channels, namely one to its parent (A-B) and/or one to its (older) 'brother' (B-C) and/or one to its first child (B-D).

Advantages of topology 2a:
- Straightforward introduction of communication channels, only two nodes are involved.
- It follows the order scheme.
- No publishing of (confidential) business process information. Two connected nodes do not have confidential information to each other. A node only sends progress messages to participants it is doing business with. If a node receives a progress message and it is also doing business with other participants then it can send a (new) progress message to nodes representing these participants. In short, a progress message only travels one single communication channel and thus no proxy-nodes exist.

Disadvantages of topology 2a:
- The distance between a sender of a progress message and a receiver using it is *O*(n).
- The higher the distance, the higher the chance of a failing node, e.g. unreachable.

Advantages of topology 2b:
- Mediocre complexity when introducing new communication channels, a maximum of three nodes can be involved. The three nodes are the outsourcing node, the accepting node and the root of the network topology.
- The distance between two nodes is *O*(1).
- The routing mechanism needed by the root is trivial.

Disadvantages of topology 2b:
- A single point of failure (the root).
- Use of resources is focused on one node.
- (Confidential) business process information is published (although limited). The root and only the root, can receive confidential business process information of other nodes.

Q508

Advantages of topology 2c:
- Limited number of communication channels per node (resources).
- Mediocre complexity when introducing new communication channels, a maximum of three nodes can be involved.

Disadvantages of topology 2c:
- The distance between two nodes is *O*(n).
- A routing mechanism is needed.
- (Confidential) business process information is published. For example, B receives progress messages from C that it has to pass through to A.

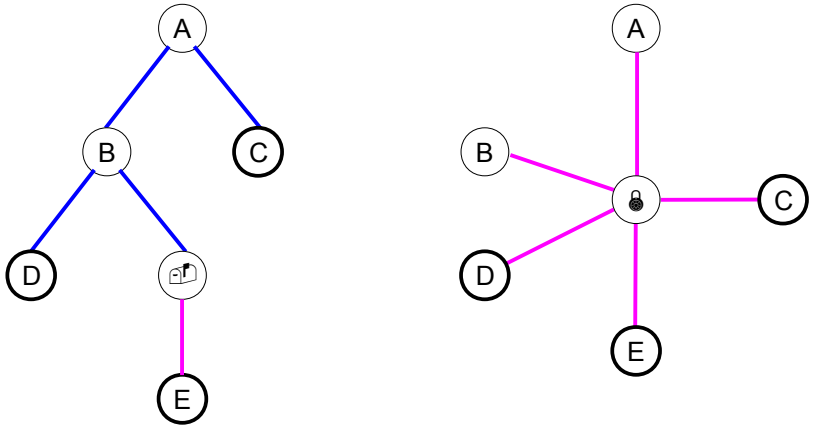
Network topology approach 3: introduction of special nodes

Q301

Instead of only introducing communication channels between nodes that represent companies it is also possible to introduce additional nodes that only function as special kind of proxy-node, called special-node. Special-nodes appear in the network topology, but not in the order tree. One of the properties of a special-node is that it can have a higher availability percentage than mentioned for other nodes, see RQP006. A special-node can also act as a buffer within a normal communication channel and can have a positive influence on the number of successful delivered messages. This can especially help when mobile nodes exist in the network that are assumed to have less reliable connectivity. All messages sent to the mobile device can be stored at the special-node and are

retrieved when a connection to the special-node is made.

Another property of a special-node is that it can create a more confidential communication network. One assumption for this has to be made, namely that the special-node is a trusted third party for all the nodes involved. The special-node can act as a pass trough node for all messages and make the promise that it doesn't read any of them.
Since a special-node is always in between two or more nodes it is not possible to create a network topology with only direct communication channels between the (non proxy-node) nodes. Topologies that can be built are variations of those of approach 2. The following two figures represent network topology 2a and 2b with a special-node acting as buffer and trusted third party respectively:



Figures 8.6.a-b – topology 3a with a buffer special-node (8.6.a), topology 3b with a trusted special-node (8.6.b)

Topology 3a is equal to topology 2a except that node E now is a mobile node that communicates through a buffer. Messages sent to node E are temperately stored at the buffer and requested by node E when it has connectivity. Messages sent by node E also pass the buffer that takes care of delivering them at the next node, i.e. node B in this case.

Every node can be replaced by a buffering special-node together with the replaced node although this can have an undesired delaying effect on the communication. A buffering special-node therefore always acts as a buffer for only one node without children. The advantage of the buffering special-node is that it improves the communication with mobile nodes. The forwarding of messages is discussed later in this chapter.

Topology 3b is a modification of topology 2b where the trusted special-node takes care of the delivery of messages at all other nodes. Since this special-node acts as a trusted party, all the other nodes assume that the special-node handles the message as confidential information. The buffering special-node of topology 3a can also act as a trusted special-node to ensure that messages from/to node E are not read. Q508

The properties of approach 1, 2 and 3 are summarized in the following table:

Property / Topology	**1a**	**1b**	**2a**	**2b**	**2c**	**3a**	**3b**
Only direct communication*	✔	✔	-	-	-	-	-
Unused communication channels*	-	✔	-	-	-	-	-
Complexity of adding nodes*	■■■	■■■	■	■■	■■	■	■■
Distance between communicating nodes**	*O*(1)	*O*(1)	*O*(n)	*O*(1)	*O*(n)	*O*(n)	*O*(1)
Complexity of routing*	-	-	-	■	■■	-	■
Single point of failure*	-	-	-	✔	-	-	✔
Publishing confidential information*	-	-	-	✔	✔	✔	-
Follows the order scheme*/***	-	-	✔	-	-	✔	-

Q007
Q508

* *- = Absent; ✔ - Present; ■ - = Straightforward; ■■ - Mediocre; ■■■ - Complex*
** *Big O notation*
*** *Nodes only communicate with other nodes that are known from placing/receiving the order*

Table 8.1 – properties of topologies 1, 2 and 3

Table 8.1 summarizes that network topologies 2b, 2c and 3a publish confidential information. This property requires some additional attention, because it is not a technological property that a transport company can consider as a problem of its technology partner (software company). Because competitors within the transportation industry can abuse it, it can be a barrier for these topologies to be accepted in practice. A solution for this problem is to introduce encryption using an asynchronous encryption algorithm. However, preventing confidential information from being published by using a specific topology is a preferred solution, because it is easier to implement and requires less resources and administration.

8.4 Message routing and forwarding

The previous paragraph mentioned the routing of messages through the network topology when a sender and a receiver are not connected by a single communication channel. This part of the chapter focuses on some basic principles of routing, well-known routing methods and applying them on the network topologies.

The basic problem of routing is how to make the decision on which output line an incoming packet should be transmitted [Doyle] where lines and packets are equivalents of communication channels and messages within this document. The routing problem occurs when a collection of nodes -or subnets within routing documentation- is connected and there exists at least one path between every two nodes [Caldwell].

There exist many methods for routing such as flooding, distance-vector routing, link state routing, hierarchical routing and multidestination routing that are described by [Doyle] and in more detail by [Tanenbaum]. The goal of these methods is to deliver a message at the intended receiver, but they all have certain properties that makes them appropriate for specific situations. This paragraph only describes the main aspects of several route methods, but more detailed description are given in appendix G.

Flooding

The main aspects of the flooding method is that if a node receives a message, it is forwarded to every other node it can directly communicate with unless this node receiving the message is the intended receiver. It supports a technology to prevent endless forwarding. This method guarantees

that the shortest path is used and a high chance of delivery, but wastes a lot of bandwidth.

Distance-vector routing
The main aspect of the distance-vector routing method is that every node constructs a table with a 'distance' to every other node where a node can appear twice (with possible different distances) when there exist multiple routes. It takes some time for every node to construct or alter a table containing every node in the network. Every record in the table includes which communication channel has to be used to send a message to the node of that record. This method doesn't waste a lot of bandwidth and guarantees that the shortest path is used, but doesn't work very well when nodes fail because of the time needed to reconstruct the distance table using alternative routes.

Link state routing
The routing method is a combination of the flooding and distance-vector routing methods. It uses flooding the spread the information required to construct the distance-vector routing table. This is a trade off between wasting bandwidth and quickly (re)building the routing table. The latter makes it better suitable for situations where a node fails solving this disadvantage of the distance-vector routing method.

Hierarchical routing
This routing method can be put on top of the two previous routing methods when the network consists of many nodes. It splits the network into several sections where the distance-vector table of each node only contains information about nodes in its section what saves resources. Additionally, every section has an 'exit'/'entry' node that connects the sections together. If an intended receiver is not in in a section then the message is sent to the exit/entry node to forward it to a correct other section. The slightly longer paths and heavy traffic load at the exit/entry node are disadvantages.

Multidestination routing
This routing method is an extension to existing routing methods by introducing functionality to let a single message contain multiple receivers. If a single message frequently has to be send to more than one other node then this routing method decreases the total amount of messages sent (heavily). Together with another route method that doesn't waste bandwidth, it facilitates a network with as low bandwidth usage as possible.

Consequences of the network topologies
Before being able to take a look at routing methods suitable for the different topologies, one important aspect has to be pointed out. The described routing methods all assume that a network exists of many communication channels that usually form more than one path from one node to another. However, all the topologies mentioned only have one path from one node to another because they are all trees. This has the following effects on the described routing methods:

Flooding: no cycles exist and thus it is not possible to receive a message twice. The technology to stop the message from being flooded is therefore not needed, because it will stop at all the leafs of the tree. There exists only one path between two nodes and so the the advantage of a high chance of delivery be ignored.

Distance-vector routing: the knowledge about the distance to all nodes using a specific communication channel makes it possible to create a so called Sink Tree with shortest paths from one node to every other node. The network topology (a tree) created by the building steps is therefore automatically the Sink Tree of all nodes. With no alternative routes available, the 'node failure' problem doesn't exist.

Link state routing: the effects of the previous two routing methods also applies here. The more detailed description in appendix G mentions the problem of aging. This problem doesn't exist, because there exists no alternative paths and thus will the flooded routing information after a

reboot be equal to that before the reboot.

Hierarchical routing: the effects of the previous routing methods also apply here. One could question whether this method is of any use in practice, because very large network of nodes probably don't exist when it comes to outsourcing.

Multidestination routing: there is no effect on this routing method. It can be an (valuable) addition to the previous three routing methods.

8.4.1 Routing within network topologies

The previous paragraph described several routing methods that can be used to deliver messages at their intended receiver. Now it is possible to take a look at the network topologies and their most suitable routing method.

Extending topology 1a, 1b and 2a with routing

Topologies 1a, 1b and 2a don't need any routing method, because they only use direct communication channels. The first two topologies have a direct communication channel to every node they have to send progress message to while the third topology only has communication between parent-child pairs that are always connected due to the order placing-accepting relationship.

Extending topology 2b with routing

Topology 2b is a star network with the root as central node. This implies that the maximum number of communication channels a message has to travel is limited to 2, namely from the sender to root and from the root to the destination. The only node in this network topology that has to route messages is the root.

There is only one routing method that is really suitable for this topology, namely distance-vector routing. This method guarantees the (trivial) shortest path and wastes no resources. Although flooding is easy to implement and will work, it wastes a lot of resources that is clearly worthless. Link state and hierarchical routing are not suitable because only one routing node exists.

Extending topology 2c with routing

This network topology focuses on limiting the number of communication channels of every node to a maximum of three. The characteristic of this topology is that it creates relatively long chains of nodes wherein a node can only route a message to its only other communication channel. This implies that the flooding method doesn't necessarily wastes as much resources as topology 2b does, because a destination can be reached before it is flooded to the end of the string. This makes flooding a suitable routing method for this topology although wasting of resources still occurs.

Distance-vector is a suitable routing method, because it guarantees the shortest path. It is more suitable then link state routing, because no alternative paths exist while it uses more resources than distance-vector routing. Hierarchical routing can not create longer paths, but is less suitable due to the practical aspect.

Extending topology 3a and 3b with routing

Topology 3a and 3b differ from 2a and 2c in the addition of a buffering proxy-node and a trusted proxy-node. A buffering node does not influence any routing method described, because it only acts as a (more reliable) replacement for the original node. This means that the buffering proxy-node takes over the routing of the node it replaces and that this replaced node doesn't need routing functionality. This is an advantage because messages that pass the buffering node do not have to wait until the mobile node has connectivity. Apart from a buffering node that doesn't

influence a routing method, topology 3a doesn't have any routing at all.

The trusted proxy-node fulfills the same role as the root node in topology 2b. This implies that the trusted proxy-node is the one that routes the messages between all nodes and adds the guarantee that it doesn't read the contents of the message. The flooding routing method is not suitable anymore, because messages are delivered at all nodes, including those that are not trusted.

All topologies have one ore more suitable routing method to provide the functionality for requirements RQP001, RQP003 and RQP005. The topologies together with one of the routing methods create a communication networks that can be used to send progress message from one node to another and is described in the next part.

8.4.2 Progress messages to downstream nodes

Requirement RQP002 mentions the ability for a node to know the nodes that it has to send progress messages to. This collection of nodes can only be derived from the flow of goods scheme. Unfortunately, nodes only know their parent and their child node(s), because these are the nodes where they placed or received their order from. Even in topologies that don't follow the order tree, such as 2b and 3c, a node only has a communication channel with another node of which it doesn't know its role.

An important aspect in sending progress messages is the limited knowledge of nodes about the other nodes in the order tree. A part of knowledge that every node has, is the collection of children it outsourced (a part of) the order to. Of course it is possible that a node does not outsource. This knowledge can be used by every node to create an execution order on the children responsible for a part of its transportation order as well as its own transport (if it does any). For example, in use case 4 company A knows that B and C are both responsible for a part of the transportation where B appears before C in the execution. A doesn't know whether B or C also outsourced a part of their order, but it does know that if this is the case then the children of B appear before C. This implies that progress messages created by nodes in the C-subtree are not needed by the nodes of the B-subtree. However, they are needed vise versa. From this it can be concluded that nodes of the C-subtree would like receive progress messages sent by nodes in the B-subtree.

Q004
Q406a

The following figure illustrates the execution order of nodes in the order tree of use case 4 where node B is explicitly extended with executing a part of the transport itself between that of node D and E. The execution order is indicated with blue numbers for every node.

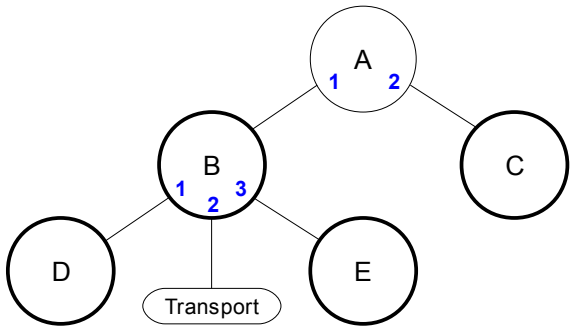


Figure 8.7 – execution order within altered use case 4

The question that needs to be answered is: what algorithms can take care of delivering progress messages at specific (interested/intended) nodes by using the limited knowledge of a node and using the properties of a network topology?

First, the network topologies can be categorized into two groups:

Topology group one: the topologies in this group follow the order scheme (2a and 3a). This group is referred to as the first group.
Topology group two: the topologies in this group don't follow the order scheme (all except 2a and 3a). This group is referred to as the second group.

Q406a
Q406b

The first group is assumed to be more acceptable in practice, but the second group introduces some technological advantages.

General aspects of both groups

The first general aspect is that a parent always receives progress messages from its children. A parent can also send progress messages to its children.

The second general aspect is the limited knowledge about the order tree existing at a node. As mentioned earlier in this paragraph, a node is able to put an order on the execution by its children and itself. The progress messages that a node would like to receive are those sent by other nodes 'on the left' in the order tree as well as those of its children. To illustrate the addition of a node, use case 4 is used before the addition of D and E:

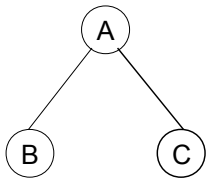


Figure 8.8 – use case 4 before addition of D and E

Initially, A receives progress messages of B and C, because they have an outsourcing/accepting relationship (RQP001) and it is possible for A to send progress messages to B and C. Starting from this situation there are two possibilities where a new node can be added, namely as a child of B or as a child of C. If the new node is added as child of B then all the nodes 'on the right' would like to receive progress messages, but the only node that receives them is node B. If the new node is added as child of C then the new node would like to receive progress messages from nodes 'on the left', but the only node that receives progress messages is node C (coming from the new node). The addition of a new node in both groups always has two goals, namely:

- The new node would like to get progress messages from nodes 'on the left'.
- Nodes 'on the right' would like to receive progress messages sent by the new node.

Specific aspects of the first group

The most distinguishing aspect of the first group is that a node only sends progress messages to its parent and/or its children. In other words, a node only sends progress messages to nodes it is directly connected to in the order tree (and thus in the network topology). If a parent, that receives a progress message, also has a parent (multi level outsourcing) then this parent also has to send a progress message, what is a new message and not a forward of the message received from the child.

The following algorithm or rules take care of the sending and receiving or progress messages.

- A new child that is added to the order tree doesn't send any 'announcement' message because its parent already knows about its existence and no other nodes need to know about its existence.

- If a node creates a progress message then it sends a progress message to:
  - its parent (if it exists) because it is obliged to
  - its children (if they exist) of which it knows that they have a higher execution order ('on the right')

- The created progress message now goes upwards (to the parent) and downwards (to the children) in the order tree. Inverted, it is received from from beneath or from above. If a progress message is received from beneath then the node sends a progress message to:
  - its parent (if it exists) because it obliged to
  - its children (if they exist) of which it knows that they have a higher execution order than the one it received the progress message from ('on the right')

- If a progress message is received from above then a progress message is sent to all the children (if they exist). The fact that the message was received from above indicates that the receiver is 'on the right' of the sender. The sender doesn't even know whether the receiver has children.

- This algorithm can be illustrated with use case 4. When node D sends a progress message then the information travels upwards to the root. Node B and the root both send the information to children 'on the right', which are E and B.

Specific aspects of the second group

If a node would like to receive the progress messages, a solution can be that it is able to take a subscription on the progress messages sent by nodes that appear earlier in the flow of goods scheme. This means that communication can appear directly between other nodes than just those in the order outsourcing/accepting relationship. Two requirements to enable those subscriptions are that a new node has to inform the others about its presence and that a node that receives the announcement initiates the right actions to enable a subscription.

The first requirement involves two goals, namely get a subscription on the progress messages of nodes appearing earlier in the flow of goods scheme as well as telling other nodes to take a subscription on its progress messages if the appear later in the flow of goods scheme. The right actions of the second requirement can be summarized as a set of rules that a node has to follow when it receives an announcement. Before introducing these rules, the three kind of messages required by the rules are introduced, namely:

- Announcement: telling receivers that it is part of the order tree and is executing a part of the transport. The message contains the identifier of the new node. Announcement messages are not addressed to a specific node.

- Subscription request: requesting a subscription to the progress messages of a specific node. A subscription request is addressed to a single node.

- Subscription suggest: telling receivers that a new node announced itself and appears earlier in the flow of goods scheme. The message contains the identifier of the new node. Subscription suggest messages are not addressed to a specific node.

The following rules introduce an algorithm that takes care of the two goals, but first some assumptions are made to simplify the algorithm and to ensure that all companies appearing in the order tree can rely on the same expectations:

- If a node receives a subscription request and it is performing a part of the transport itself then the requesting node is added to the subscription table.
- A node that actually executes a transport sends progress messages to the subscribers of its subscription table.
- Superfluous: all nodes obey the rules below.

Rule 1: introduction
Every new node that is going to execute a part of the transport has to introduce itself by sending an announcement to its parent.

Rule 2: receiving an announcement from a child
If a node receives an announcement from a child then it knows that a node is added somewhere in the subtree below. The receiving node has to:

1. Forward the announcement to its parent (if it has any)
2. Forward the announcement to all of its children (if it has any) with a lower execution order than the one it received the announcement from.
3. Send a subscription suggest to all of its children (if it has any) with a higher execution order than the one it received the announcement from.
4. Add the announcing node to its subscription table if it executes a part of transport itself with a lower execution order.
5. Send a subscription request to the new node if it executes a part of transport itself with a higher execution order.

Rule 3: receiving an announcement from a parent
If a node receives an announcement from a parent then it has to:

1. Forward the announcement to all of its children (if it has any).
2. Add the announcing node to its subscription table if it performs a part of the transport itself. This is needed because the announcing node appears later in the flow of goods scheme (derived from rule 2.2).

Rule 4: receiving a subscription suggest from a parent
If a node receives a subscription request from its parent then it knows that it appears later in the flow of goods then the added node. Due to rule 2.3 a suggest is only sent by a parent to its children (downwards in the order tree). The receiving node has to:

1. Forward the subscription suggest to all of its children (if it has any)
2. Send a subscription request to the sender if it executes a part of transport itself.

The rules are illustrated using the following complex order tree:

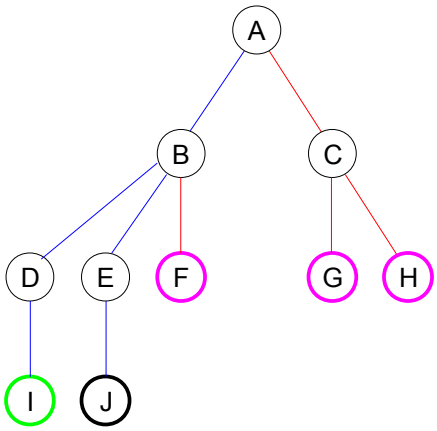


Figure 8.9 – subscription requests using the four rules

Node J is just added to the order tree. The blue edges indicate that an announcement has been

sent from one node to another. The red edges indicate that a subscription suggest has been sent from one node to another. A green node indicates that this node has added node J to its subscription table. A purple node indicates that this node has sent a subscription request to node J.

Due to rule 1, node J sends an announcement to node E. Next, after applying rule 2.1 two times, the announcement is received by node B and A. It is also received by node D and I because of rule 2.2 and 3. In short, the first three rules take care of delivering the announcement at all nodes up to and including the root (1 and 2.1) as well as to all the nodes 'on the left' (2.2 and 3.1). Now I receives progress messages from all nodes on 'on the left' and those up to the root that execute a part of the order before the new node (2.4, 2.5 and 3.2).

Rule 2.3 takes care of the first step of sending subscription suggestion messages to all the nodes 'on the right' and is applied by node B and A. Finally rule 4 takes care of forwarding the message to the other nodes 'on the right' and requesting subscriptions (4 and 2.5).

In short, the algorithm splits the order tree into a blue subtree on the left of the root and into a red subtree on the right of the root. The nodes of the red subtree added the new node to their subscription tables. The nodes of the blue subtree sent a subscription request to the new node. The nodes that appear on the path from the new node to the root (blue colored), and also perform a part of the transport, are taken care of by rule 2.4 and 2.5.

8.5 Summary

This chapter describes three reasons why communication between participants is needed, namely to get quotations, place orders and acquire progress information. A communication network between the participants can be created based on three network topology approaches. These three approaches distinguish their selves on using direct and indirect communication channels as well as the requirement of special addition communication nodes that provide support for unreliable connectivity and more secure communication. Some of the network topologies are suitable for situations in which communication can only appear between an order placing/accepting pair of nodes.

All communication network alternatives have a different score for the complexity of adding nodes, the distance between nodes, the requirement of routing and the publishing of confidential information. There exists a correlation between the complexity of adding nodes, the distance between nodes and the requirement of routing functionality (of which several alternatives are given). There also exists a (trivial) correlation between the property of direct communication and the publishing of confidential information. None of the alternatives can be assumed to be a best solution to create a communication network, because this depends to heavily on the importance of one or more desired properties. However, all alternatives enable communication between all nodes that would like to communicate because of the three reasons.

The third reason describes the need to send and receive progress information between nodes that can appear before or behind each other in the flow of goods scheme. For every topology an algorithm is given for a node to be able to know to which nodes it has to send progress information to. This collection of nodes is referred to a as the nodes appearing in the progress information subscription list.

Knowing that it is possible to create a communication network that is suitable for all three reason of communication, it can be used as a communication layer that, in its turn, can be used by an application that actually implements the the three reasons. The next chapter focuses on creating an application that implements several business processes of chapter 6 using the CDM of chapter 7 and based on a communication network being available from this chapter.

9 Exchanging progress information

The previous chapter describes three reasons why companies have to be able to communicate with each other. This results in several options to create communication networks that can be categorized into two groups, namely one that follows the order tree and one that doesn't. For both groups, a communication network, based on one of the topologies of the previous chapter, is assumed to exist.

The first two reasons only involve several messages between an order placing/accepting pair of nodes of which the messages can easily be derived from an existing solution, for example RosettaNet. Therefore, this chapter focuses primarily on the the third reason for communication, i.e. sending and receiving progress information, with the differences between the two groups in mind. This chapter combines business process 'Execute order' (chapter 6) with the Common Data Model (chapter 7) having the assumed communication network.

After focusing on the exchange of progress messages, this chapter takes the growing and shrinking of an order three into consideration. Next, a closer look is taken at specific rights that companies need to have to alter information that is used by multiple companies. This subject has already been mentioned within paragraph 6.5 as one of the aspects that should be considered when business processes cross company borders.

9.1 CDM applied to outsourcing

Chapter 7 described a Common Data Model that can be used by transport companies to store information about their transport processes. Another option is to create the possibility to translate information to and from the CDM using an existing TMS. The previous paragraph mentioned two reasons for sending progress messages, namely:

- To hold a company accountable
- To inform downstream companies about the transportation progress.

This paragraph focuses on the influence of the progress messages on data stored using the CDM. First, the primary elements of the CDM are repeated to take a look at how orders are stored. This is illustrated by figure 7.1 of paragraph 7.2.1. The main content of an order is given by transportables that describe goods including their physical aspects, origin and final destination. For example, an order can consist of three boxes that are described by three transportables. When the transportables do not need to be transported from the origin to the final destination then the track over which they have to be transported by the transport company is described by a TransportableTrack that refers to a single transportable. This implies that when all three boxes of the example need to be transported over the same track, there exist three TransportableTrack items that refer to each transportable.

To keep track of the history of a transportable, i.e. to keep track of a box in the example, the CDM contains the LocationMoment entity. Every instance of a LocationMoment refers to a single transportable and includes information about the location where a transportable was at a certain moment. Another entity, namely Holder, describes the long-lived resource or building in which a transportable is contained, for example a sea container, vehicle or warehouse. A LocationMoment instance can have a reference to a holder to give more information about the location. The following figure illustrates an example of the contents of the CDM entities when an order is received to transport a single box. The details of the relations as well as the possible reference from a LocationMoment to a Holder are left out for simplicity.

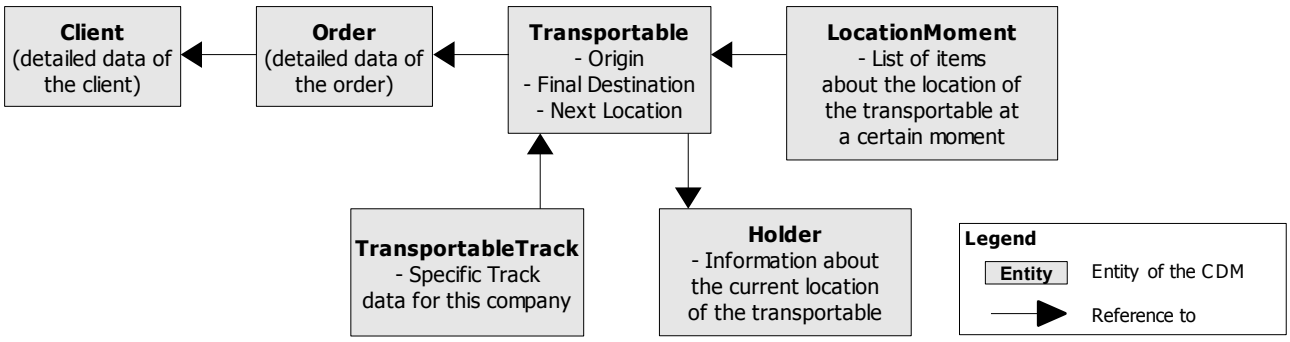


Figure 9.1 – simplified CDM order representation

Figure 9.1 can be further simplified by leaving out the Client and Order entities, because the client is trivial and this chapter assumes that only one order exists at each transport company accepting the order. This implies that the transportable always has a reference to that single order. Another assumption that is made, is that each order only consists of one transportable, because multiple transportables only make figures and descriptions more complex, while all other details remain the same.

The following figure displays figure 9.1 without the Client and Order entities as well as instances of the entities using the following example:

- A box needs to be transported from ABC in the Netherlands to KLM in Belgium.
- Physical aspects are not part of the example for simplicity.
- A warehouse exists at the Dutch/Belgium border and is called NLBE.

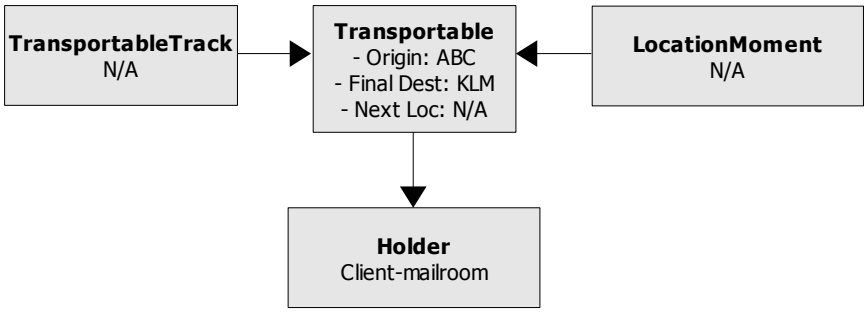


Figure 9.2 – more simplified CDM order representation

Paragraph 7.3 describes additions to the CDM to enable it for outsourcing. If a transport company outsources (a part of) an order then the CDM provides OutsourceTrack entities for the administration of outsourcing. The transport company in the example (A) places an order at two other companies (B and C) that both take care of a part of the total transport. This implies that company A uses two instances of OutsourceTrack and that company B and C both have an instance of TransportableTrack to describe their track of the total transport. The track of company B is from ABC to NLBE-storage and that of company C from NLBE-storage to KLM. To be able to put all the entities in one figure, the relations are left out and the added arrows represent outsourcing. This results in an order tree with CDM information where companies are represented by nodes.

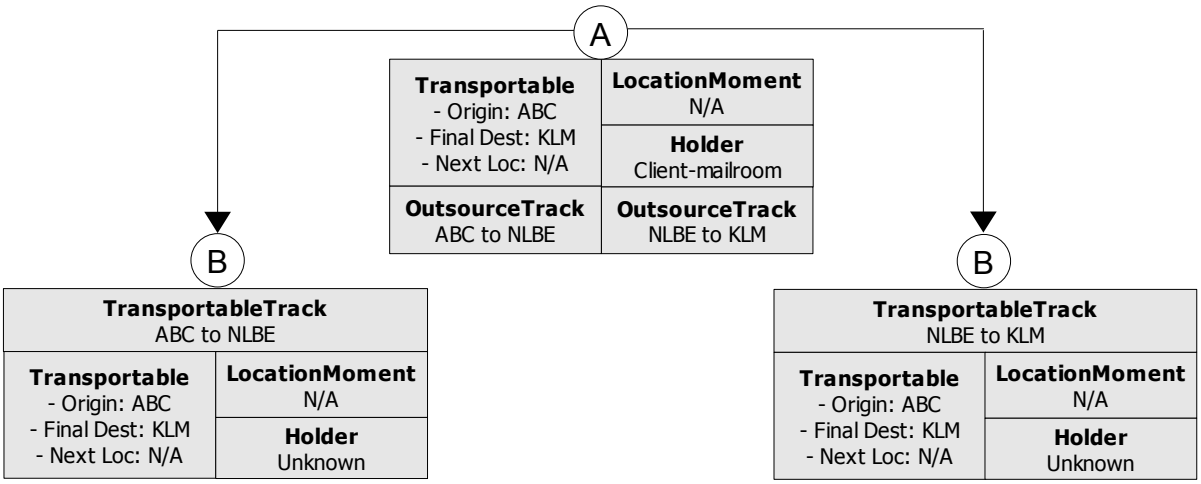


Figure 9.3 – CDM integrated into the order tree

The figure above displays the nodes and their data using the CDM after node A outsourced its order partially to node B and C. Several properties can be derived from figure 9.3, namely:

- The data of a transportable is equal for all nodes
- The OutsourceTrack instances have an equality with the TransportableTrack instances of the companies outsourced to.
- None of the companies has instances for LocationMoment, because the transport hasn't started yet.
- Only company A knows the holder.

All involved companies have the data required to execute their part of the transport and if no problems occur during the execution then the order will be completed successfully with both nodes B and C having Proofs of Execution. However, when these nodes do not inform others about the transportation progress and/or possible delays one cannot be held responsible or react to possible delays. These problems are exactly the two reasons why progress messages exist. The next paragraphs focus on these two reasons and how the progress messages reflect on the CDM.

9.1.1 Progress messages and the CDM

First, the reason to hold a company accountable is reflected on the CDM to get a view on the information changed or added when a progress message is received. To be able to do this, it has to be defined what information is required in a progress message. In other words, what information is required to hold a company responsible? Another question that can be asked is whether this information is different within the two groups. The following information can be used to hold a company responsible:

Q004

- The current location of the transportables. This information gives an answer to the clients' question 'where are my goods?' and can be considered as basic Track&Trace information.
- The locations where the transportables have been in the past. This information can be used to judge possible delays or other negative occurrences.
- The proof of execution information. This information contains a name, signature and time of a sender of receiver at specific location.

This information enables a customer, whether it is a client or an outsourcing transport company, to know the current and past locations of its goods as well as pick-up/delivery information. This information is assumed to be enough to hold someone responsible in case something goes wrong. Because sending progress information to a parent is obligatory in both groups, there is no

difference between the group that follows the order scheme and the one that doesn't.

The three information elements mentioned are present within the CDM. The current location of the transportables is represented by the holder that provides a location. The history of locations where the transportable has been is provided by the LocationMoment items. This entity represents a location, including the point in time, where a transportable has been. The proof of execution attribute is available in both TransportableTrack and OutsourceTrack.

When a company is executing its part of the transport it can change the holder and/or its location to store the current location. An item of LocationMoment is typically added when the holder changes. The proof of execution can be registered by changing the attributes of TransportableTrack. The task of the progress messages is now defined as taking care of updating and/or adding the following information at a parent node:

- If a node changes the holder then this change has to be send and processed by the parent.
- If a node adds an item to LocationMoment then this item also has to be added to LocationMoment by the parent.
- If a node completes a task and stores a proof of execution by changing the TransportableTrack then the parent also has to make this change to its OutsourceTrack.

In short, it can be said that the data of Holder, LocationMoment and TransportableTrack / OutsourceTrack has to be equal for both nodes. The data of the Transportable can be added to this list to ensure that possible small changes to it, such as minor corrections, are also known by both nodes. Updates and/or additions to the data can be send from one node to another using progress messages, but this introduces the following problems:

- If a node sends a progress message to another node then how does this sending node know that it has been received and processed successfully by the intended receiver? This problem can occur when hardware/network failures appear or when acknowledge messages are absent.
- If a node sends multiple progress messages in a short time interval, how can the receiver process them in the correct order?
- It is not excluded that both nodes make changes to the same attribute, such as a minor correction of the data of a transportable, at the same time. How is this conflict resolved?
- If data is considered to be equal at a child-parent pair of nodes, how do changes affect the data of other children of the parent?

The assumption of data being equal at multiple nodes and the problems described above are typical assumptions and problems within the topic of distributes databases [Camarinha]. For now, it is assumed that the techniques for distributed databases can be used to solve the first three problems; the topic of distributed database techniques is therefore discussed later on in this chapter.

Q504

The last problem can be illustrated by considering the LocationMoment items that refer to a transportable. If these items are equal for nodes A and B then so are they for A and C. This implies that, when B is executing its order, C can follow its progress, by examining the LocationMoment items, although this information is not required by C to be held accountable by A. In fact, it can publish confidential information of node B to node C. A solution to this problem is not given in detail, but additional attributes to the LocationMoment entity can split the items of LocationMoment between involved nodes.

Q007

Progress messages and downstream nodes

The second reason for process messages is to inform downstream nodes about possible delays or other unforeseen issues. Every downstream node executes a part of the whole transport. In other words, all the partial transports of all nodes together form a track from the original location to the

final destination and are thus required to be interconnected. This implies that all the TransportableTrack instances, as described in the previous paragraph, have the property that the delivery location of one node is the pick-up location of the company that executes the next part. This is also true for OutsourceTrack instances, because these have an equality relation with a TrackportableTrack of a node the order is (partial) outsourced to.

This paragraph focuses on delays that can occur during the execution of the transport. The term delay is defined as an event during the execution of a transport that influences the earlier estimated time of arrival of a transportable. A change to the estimated time of arrival can have influence on the planning of other (downstream) transport companies. Naturally, the client will probably not be satisfied with any delay, but this is out of the scope of the execution of the transport. The contents of a progress message to inform downstream nodes about a delay can be one of two kinds, namely the actual delay, that is a delta, or a new estimated time of arrival, that is absolute.

The network topologies topic made the distinction between two groups when it came to sending progress messages to downstream nodes. These two groups were:

1. Network topologies that follow the order tree.
2. Network topologies that don't follow the order tree.

This paragraph takes a closer look at the progress message used inform to downstream nodes about delays and also makes the distinction between the two groups.

Delays and progress messages in the first group

The primary aspect of the first group is that communication only appears between two nodes, namely a parent and one of its children. If a delay appears during the transport executed by the child it has to send a progress message to the parent to inform it about the delay. The parent in turn can send a delay message if it is a child of another parent. This process can continue up to the root of the order tree, but the main aspect is that all progress messages are new (fresh). The rules described in paragraph 8.4 also describe other events of sending a progress message in scenarios.

The previous paragraph mentioned two kinds of content for delay progress messages, namely a delta value and an absolute value. The steps that are taken are as follows:

1. A delay occurs at the child
2. The child creates a progress message (delta or absolute)
3. The child sends the progress message to the parent
4. The parent receives the progress message
5. The parent changes the estimated time of arrival of the OutsourceTrack
6. The estimated time of arrival is also changed at the childs' TransportableTrack

The last step is based on the equality of an OutsourceTrack and TransportableTrack described earlier. This aspects introduce an alternative to inform the parent about a delay, namely by changing the TransportableTrack at the child which implies that the OutsourceTrack is also changed accordingly at the parent. The steps above suffer the same problems as the progress messages sent for being accountable such as network/hardware failures. The alternative is based on the idea of a distributed database that is assumed to solve these problems.

The next step is take a look at what happens at the parent after its OutsourceTrack has changed. If node B of the example has a delay then this can influence the execution of node C, but this doesn't necessarily need to be true. For example, if the delay is 1 hour and the the TransportableTrack of node C described loading the transportable 2 hours after delivery at NLBE then no problem exists, but it does if the delay is 3 hours. Thus, if a delay hasn't got any influence on the remaining track the parent simply notifies the delay, but doesn't need to take any further action (a solved delay).

This makes the process of informing downstream nodes about a delay somewhat intelligent compared to just simply altering all times of OutsourceTrack/TransportableTrack instances.

It is also possible that a delay does influence the transport of at least the first downstream node. For example, the estimated time of arrival of B at NLBE is later than the planned loading of the transportable by C. In other words, after the OutsourceTrack is changed at A, an overlap appears in the time windows of both OutsourceTrack instances. This situation is now referred to as a high priority delay. The only possibility to solve a high priority delay by A is to postpone the times of the OutsourceTrack of C. At least the moment that C can load the transportable needs to be postponed, because the transportable is physically absent at NLBE at the original time. Postponing the track is done by A by simply changing the OutsourceTrack describing the transport of C, because this also changes the TransportableTrack of C.

From the delay situation described above, it can be concluded that the algorithm or rules that apply to the first group for sending progress messages in fact describes what OutsourceTrack or TransportableTrack instances need to be changed. The steps that are taken when a (high priority) delay occurred are equal for all nodes in the order tree. Therefore a high priority delay at the start of flow of goods scheme can involve a lot of changes to OutsourceTrack or TransportableTrack instances, but at least the algorithm takes care of updating all downstream nodes.

Although progress messages in the first group can successfully inform all downstream nodes about a delay, one assumption seems to have been made, namely that any node simply accepts any change to its TransportableTrack. In practice this cannot be true, for example when a transport company has no vehicle available at the new times. The only solution to this problem is to cancel the order which implies that the parent has to place a new order of its OutsourceTrack at another transport company. The question which company has what responsibility and possible (financial) consequences are legal issues and are out of the scope of this thesis.

Delays and progress messages in the second group

One of the assumptions for progress messages in the second group is that every node has a subscription list that contains all downstream nodes, see paragraph 8.4.2. In comparison to the first group, a node is able to send progress messages to all downstream nodes in parallel where in the first group a child only sends a progress to its parent. The variants of the content of a progress cannot be equal to those of the first group. It is not possible to send an absolute time to all nodes of the subscription table, because the receiving nodes do not know the old time and are therefore not able to calculate the delay, but it is still possible to send the delay as a delta. However, this method suffers the same problems as the progress messages sent for being accountable, although it is simple and straightforward. Since it is assumed that distributed database technology solves these problems, changing an instance of an entity that is known by all downstream nodes will create a solution.

Within the first group, TransportableTrack instances are changes by children to inform parents about a delay. Unfortunately, this is only possible between a parent and a child node, because these instances are not 'shared' with downstream nodes. The properties derived from figure 9.3 however indicate one entity whose instances are equal at all nodes in the order tree, namely Transportable. This implies that the only solution to inform downstream nodes using distributed database technology is by making changes to the instance of Transportable.

The previous paragraph about the progress messages to downstream nodes in the first group described a kind of intelligence that created a difference between easily solvable delays and high priority delays. It is possible to have this same kind of intelligence within the solution for the second group by examining the knowledge of every node about their TransportableTrack. It is known that the delivery location of a TransportableTrack is equal to the loading location of the TransportableTrack of the next downstream node. From this it can be concluded that a node does

have knowledge about the TransportableTrack of the first downstream node. If a node is able to change the data of this TransportableTrack by making a change to the transportable then this would simply be sufficient. The attribute used to inform the next downstream node is NextLocation of the Transportable entity using the following steps:

1. Every node that starts executing a transport changes the NextLocation to the delivery location of its TransportableTrack including the estimated time of arrival.
2. This change is propagated to all other nodes, because this data is equal at all nodes.
3. If the NextLocation of the transportable is equal to the start location of a nodes' TransportableTrack then this node is able to conclude whether there exists a (high priority) delay.
4. If a node concludes that a high priority delay exists then it can change its TransportableTrack

The last step that changes a TransportableTrack can involve the same legal problems as with the first group and is therefore also out of scope. From the steps above it can be concluded that sending progress messages to downstream nodes in the second group can be done by making a change to the NextLocation attribute of the transportable. Only one, namely the first downstream node, can react to this, but it will inform other downstream nodes when it changes its TransportTrack. The second group therefore also uses the method of sending progress messages of the first group. This can be left out of the method for the second group, but then delays will only be communicated to one downstream node ahead through the change of NextLocation.

The method of sending progress messages to downstream nodes within the second group has the disadvantage that it publishes possible confidential information. This confidential information exist of the NextLocation that is known by all nodes, but only important for the first downstream node.

9.1.2 Similarity with distributed database systems

The methods for sending progress messages mentioned several problems that are solved by using distributed database technology. It is not excluded that some (if not all) problems can be solved by introducing additional technologies that, for example, ensure message delivery. However, the existence of data required and/or 'shared' by multiple nodes resembles in such a way that the use of technology for distributed databases with data replication is trivial and thus are other solutions not considered.

The network topologies of both group give a good view on how distributed databases can be used. In the first group there exists only communication between a parent and its children. This suggests that synchronization also only appears between a parent and its children. This is true when the scope of the synchronization is bound to one parent and its children. However, use case 4 consists of two parents of which one is also a child. The order tree of use case 4 is now repeated to illustrate the synchronization in the first group.

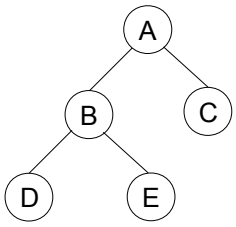


Figure 9.4 – order tree of use case 4

All nodes of use case 4 have information about the transportable that is being transported and is

equal for all nodes. For simplicity, it is assumed that the LocationHistory is also equal for all nodes, ignoring the problem of possible publishing confidential information as described earlier. Besides the data of a transportable, each parent-child pair also has equal information about the transportation track due to the OutsourceTrack-TransportableTrack combination.

All nodes having equal data about the transportable seems less complex than it is, because this equality is based on synchronization between all the parent-child pairs. This means that if node D makes a minor change to the transportable data then this is only synchronized with node B. Node B in its turn has to synchronize its changes to node A and E, et cetera. This is referred to as the propagation of the update (change). Two or more nodes that agree on an update all commit it, in analogy to a commit in a regular DBMS. Within the first group there are two ways of propagation possible, namely:

- Local-commits: this is defined as a node that propagates an update to its parent and/or children and then all commit. The term 'local' refers to the fact that the update is only propagated to the nodes an updating node has a direct connection to.
- Tree-commit: this is like a local commit, but a node that created the update or received an update only commits after it has propagated the update to its (other) connected nodes and received a commit from all of them. The term 'tree' refers to all nodes in the tree that commit the single update.

A local-commit can easily be illustrated by an update made by node D. This update is propagated to node B and both nodes commit. Now B propagates the update to node A and E, where after all are assumed to commit. Finally A propagates the update to C and both commit resulting in the update being propagated to all nodes.

The typical difference of a tree-commit compared to a local-commit is that the update is first propagated to all the nodes in the tree and then committed. This can be illustrated again by an update made by B. It propagates the update to node B. Now, B doesn't initiate a commit, but propagates the update to A and E, followed by A propagating the update to C. Nodes C and E both don't have connections to other nodes and initiate to commit. The commit of C results in A to commit. This goes on until B commits the update to D and all nodes agree on the update.

A major disadvantage of local-commit is that, during all the sequential local-commits, nodes are working on different data they assume to be correct. This causes problems when multiple nodes perform an update concurrently and two sequential local-commits can collide creating a split of nodes agreeing on a value. On the other hand, local-commits is not complex compared to tree-commit and the mentioned disadvantage doesn't occur when only two nodes appear in the order tree, because it is assumed that at the start, for a propagation to start successfully, a local transaction at both nodes is started on the object being changed.

The major disadvantage of the local-commit doesn't occur with tree-commit, because all nodes first agree on a specific update and then commit it. This can be compared to a transaction that is distributed over all nodes, but requires a lot of steps and communication. Unfortunately, the network topology only prescribes directly connected nodes to be able to communicate. This problem is solved by the network topologies of the second group in which all nodes are able to communicate with each other on the application layer. If a node would like to propagate an update then it simply initiates this at all nodes in parallel preventing a lot of waiting and saving communication costs.

It is possible to add some 'tweaks' to local-commits in the first group to resolve possible conflicts when a collision occurs. One of those tweaks can be to assign an 'always right' label to exactly one node in the tree. In case of a conflict, this node defines the correct (new) value. A typical node that this can be assigned to is the root. This adds the advantage that a propagation can be committed after the root agrees on the update and thus speeding up local-commits.

Aspects as 'always right', parallel updates and commits and conflicts are all subjects of distributed database systems, namely 'Masters', eager replication and conflict preventing/resolving. The focus on updates of transportables and OutsourceTrack / TransportableTrack in the order tree can therefore be changed to a more general focus by considering the existing technologies on this subject. The next part of this chapter takes a closer look at the different technologies for distributed databases.

9.2 Introduction to distributed database systems

Before focusing on the different kind of technologies for distributes databases, first a definition is given [ke7]:

A distributed database is a database that is stored in more than one physical location. Parts (Partition d/b) or copies (Replicated d/b) are physically stored in one location and other parts or copies are stored and maintained in other locations.

The definition points out two kinds of storing the database at more than one location, namely partitions and copies.

Parts

The first kind is partitioning the data of a table over the multiple locations. This can be done in two ways, namely horizontally and vertically. Both ways are illustrated by splitting a huge database table, namely one that has many columns and many records, over multiple locations.

Horizontal partitioning

The table (or relation) has the same columns on every location, but the records of the table are partitioned over the different locations. This means that the partitioned data of the huge table can be reconstructed by combining all the records of all locations. An example of this way of partitioning is one database location for every province of The Netherlands that stores information about the income of the citizens of that province. One advantage of this way is that if one's income changes, it only has to be changed in the database partition of that province. By combining the partitions it is still possible to calculate, for example, the average income of all citizens of the Netherlands.

Vertical partitioning

This way of partitioning partitions the huge table by storing some columns at one location and others at other locations. This implies that the number of records of every partition is equal and that every row requires a unique identifier that is part the rows of every partition. The latter is needed to be able to reconstruct the huge table, because this can't be done without knowing what part of a row of one location is related to what other part at another location. One advantage of vertical partitioning is that the partitions can be defined in such a way that the partition on each location consists of the columns that are frequently used at that location, but not at other locations. For example, if the huge table consist of columns for financial (income, total savings, unique_id) and medical (unique_id, blood type, weight) information about citizens then the financial columns can reside at the banks location where the medical columns reside at the hospitals location. Although partitioned, the government is still able to combine all columns to fulfill its role as big brother.

Horizontal and vertical combined

The two ways of partitioning the table can be combined to create a horizontally and vertically partitioned table. This means that the table is split in at least three parts, namely by first creating a vertical split based on the columns and a horizontal split of these parts based on the rows.

Copies

The second kind of storing a database or table at more locations is by creating multiple copies of it, also known as replication. This means that a table is available locally at all locations and users can perform actions, such as retrieving or updating data, from and to a table. Updates to a table at one location need to be propagated to the other locations. Concurrent use of the same table at multiple locations involves concurrency problems such as two users that would like to use (read or write) the same value simultaneously. A local DBMS uses transactions to solve this problem, but one can imagine that distributed transactions are more complex and are also affected by connectivity and communication cost aspects. Having multiple locations with the same table can provide a higher availability and capacity, because if one node fails this doesn't necessarily mean that the data of the table is completely unavailable and the database usage is distributed among the nodes. The latter implies that database systems load is distributed requiring less expensive hardware at each location.

Q301 Q303

Transparency and general advantages

Both kinds of storing data at multiple locations are assumed to be transparent to the users of the distributes database system. For partitioning, this means that if tables of multiple locations need to be combined, this is performed transparently. Another example is that if a record is removed at one location, it also is at other locations when using vertical partitioning.

For replication, transparency means that if a user reads or writes data it can assume that all data it works on is up-to-date and updates are propagated transparently. In other words, a user should theoretically not be able to recognize whether he/she is using a local DBMS or a distributed one. The word 'theoretically' here means that, for example, communication delays are ignored.

Distributed databases have several general advantages over centralized databases, namely [ke7] [WikiDD]:

- Capacity and performance: increasing the number of copies increases the capacity of total users while heavy usage of one user at one location doesn't affect the systems load for others at other locations.
- Availability: if the database system of one location is 'down' then users at other locations can still continue to use it.
- Costs: multiple small computers with a comparable capacity of one large computer are cheaper than one large computer. This also applies for the scalability.
- Localized: it is possible to physically store data near to those users that use it most. This follows organizational structures and also saves costs (communication delays).

9.3 Distributed database requirements

9.3.1 Introduction

In paragraph 9.2 it is pointed out that data of, for example, transportables is equal at all nodes. From this it is trivial that replication is the kind of distributed database that suites this property. Instead of mentioning all entities of the CDM that are considered equal in the two groups and different variants of network topologies, only the transportable entity is mentioned from now on, because other entities are analogue. This chapter will thus only focus on this entity as 'shared transportable' or 'shared data' with replication as kind of distributed database.

9.3.2 Requirements

There exist many different variants of replicated distributed database systems. In order to decide which variant is suitable for the synchronization of shared transportable data some requirements have to be defined. Using these requirements it is possible to choose the existing replication

Q301

technique to use for ELP. The requirements can be split into two categories, namely those that apply on every database system to ensure the integrity of data and those that are specific for distributed databases. To ensure integrity of the data in a database, there exist four properties that have to be met, which are also known as the ACID properties: Atomicity, Consistency, Isolation and Durability.

Atomicity: everything or nothing

The atomicity property defines that all operations in a transaction are performed or none are. It is not possible that a transaction is performed partially. This implies that database systems have to be able to keep track of what is/was going on at each moment to be able to recover from a failure that can occur at an arbitrary moment. Transactions that are not finished at the moment that a database system crashes have to be committed or rolled back during the recovery. [RQD001]

Consistency: stale reads

This problem arises when a database value is not yet updated at all nodes and the value is read (and used) by a node that has not been updated. If updates are serialized and propagated synchronously to all nodes then this problem doesn't appear. [RQD002]

Isolation: serialization and conflicts

Conflicts can be described as an event that occurs when two or more nodes update the same database value at the same time. If the update is not propagated synchronous then a conflict can appear and at least one update has to be discarded and another is kept [Dahlin et al.]. This implies that information can get lost and can be considered as not desirable. A solution can be to serialize all updates to the same database value where all updates are performed sequentially. [RQD003]

Q302

Durability: completed transactions persist

Once a transaction is committed, it is not possible to become in a state where this transaction is not considered committed due to, for example, system failures. This implies that even during a failure recovery it is known which transactions are completed in reflection to atomicity. Both durability and atomicity use persistent storage to ensure their properties by storing old values and decisions. [RQD004]

As an addition to the ACID properties, the following requirements are defined for the distributed aspect:

Support for nodes being unavailable

Due to, for example, communication problems, it is possible that a node is unavailable for some time. The network of nodes can consist of mobile nodes that are not able to communicate for some time. It is however a taken for granted that all nodes are available 50% of the time. [RQD005]

Q301
Q503

Any node can update the data

Every node that is part of the distributed database system has to be able to change the data. To be more precise, a node must be able to update or request an update of the data shared between the nodes. This implies that none of the nodes is explicitly 'read-only'. On the other hand, a rights management technique on top of the replication technique can revoke the rights of a node to update the data. [RQD006]

Simple propagation of updates

The propagation of the updates has to be relatively simple to increase the performance. It is assumed that complex propagation techniques require more communication. This has a drawback on the performance, especially if non high-speed networks are used. A higher performance can decrease the number of stale reads (if the replication technique allows this) and also decrease the time an object is locked. [RQD007]

Suitable for non high-speed networks

The network of nodes can consist of mobile nodes or other kind of nodes without high-speed communication means. High-speed is relative term because of economical and geographical reasons as well as improving communication techniques. Communication using non high-speed networks is however defined as communication using analog modems as well as ISDN modems over telephone lines. The reason for this requirement is to create a technology that is also easy accessible for companies located in more deserted areas where only telephone lines are available for communication. Another reason is that it creates a possible fall back when modern communication means such as ADSL and DOCSIS, i.e. cable modems, fail. In short, it has to be possible to use the replication system using communication means that provide a data transfer speed of 64 kilobits per second. The main difference between this requirement and RQD005 is that the latter focuses on availability while this requirement focuses on throughput; it is not excluded that a less available node (mobile node) can also have a low throughput (GPRS). [RQD008]

Real-time addition and removal of nodes

The network of nodes that share data about a transportable is built during the outsourcing of orders. This implies that the network has to be able to grow when an order is outsourced and shrinks when an outsourced order is canceled. If a transport company has completed its part of the transport then it is technically not required to be part of the network anymore. To simplify the processes of outsourcing it is now assumed that they stay part of the network. The distributed database system must support the addition and removal of nodes and therefore should not be based on a static network of nodes. The addition and removal of nodes is therefore a property that needs to be analyzed for each variant. [RQD009]

Update rights management

It has to be possible to permit updates from a node as well as revoke this privilege. One of the reasons for this is to decrease the change of faulty updates by nodes that already executed their part of the transport. This implies two properties of the distributed database system. First, it implies that there is a kind of hierarchy between the nodes that share data of a transportable. Second, it implies that there has to be a technique to grant or revoke rights to/from a node. However, this rights management technique has to be seen as an extra layer on top of the distributed database. [RQD010]

Support for removing data from the replication

The data of transportables can be considered volatile due to business process reasons. The use cases in chapter 5 describe the transport of goods from the original location to the final destination. When the goods, that are described as transportables, reach their final destination then the business process described in paragraph 6.3.3 indicates that the sending of progress messages stops. Since a progress message can change the data of a transportable, this implies that this data is not likely to change after the goods have been delivered. After a while this data would only serve for analytical purposes and there is not a real need to keep the replication in tact. To relieve the administration of the replication it has to be possible to stop the replication of data. This doesn't mean that the data is deleted at the involved nodes, only that updates are not longer propagated. The data can still be needed for future tasks, such as statistics, that would not be possible if it was deleted. [RQD011]

9.3.3 Requirements as two layers

The eleven requirements described in the previous paragraph can be used to divide the distributed database system into two layers. The first layer represents a working distributed database with a static number of nodes and no additional facilities such as rights management mentioned by RQD010. Requirements RQD001 to RQD008 are focused on this first layer. Additionally RQD009 to RQD011 are focused on the second layer that is on top of the first layer. This second layer represents additional functionality to the distributed database that is required by more ELP specific properties, such as the addition of a node what occurs when an order is outsourced. In fact, the

first layer creates a distributed database base that is required by the second layer. Paragraph 9.4 focuses on the the first layer followed by paragraph 9.5 that focuses on the addition of the second layer.

9.4 Layer 1: database replication

Replication can be described as the process of sharing data with ensuring the consistency at all participants that share this data (it is 'equal' and up-to-date at all nodes). When it possible for participants to change the data then it should be changed for all other participants that access the data. An information system that reads and/or writes the data should not have to be aware of the fact that the data is replicated, what means that the use of replication is transparent to the users of the system as described earlier.

Replicated distributed database systems use propagation to perform updates at all nodes. An update of an object has to be propagated to all nodes. Updates can be propagated using three kinds of replication techniques, namely eager, lazy and two-tier replication [Gray et al.] [Wiesmann et al.]

9.4.1 Eager replication

The primary property of eager replication is that starting a local transaction automatically involves a transaction at all nodes. This means that, just like local transactions, objects can be locked during a transaction. The three steps of eager replication are almost equal to those of a local transaction except that an update is performed at all nodes. These steps are: starting a transaction by a client, that is in fact a transaction at all nodes that can update the data, then update the object at all nodes and commit/rollback the transaction. The result is an update at all nodes or no update at all.

The propagation of the updates can be done using two update schemes. The primary aspect of the first scheme is that an update of an object can only be invoked at one node, the so called master that 'controls' the update. It is possible that different objects in one database have different masters. If a client would like to update an object then it starts a transaction at the master node. This node updates the object comparable to a centralized database using locking and logs on stable storage. After the master updated the object, but not yet committed the transaction to the client, it sends the update to the other nodes to apply it. Finally, after being sure that all nodes have applied the update using Two Phase Commit (2PC) [Silberschatz], it commits the transaction to the client. There are two situations in which the transaction is aborted, namely when the master isn't able to update the object and when the result of the 2PC procedure isn't successful. The first situation can occur because of, for example, another transaction involving the object is active while the second can occur because of a node being unavailable. The result of the update is either successful or unsuccessful at all nodes. This update scheme is called 'eager replication with master updates'.

The second update scheme differs mainly from the first one because of the possibility to update an object at all nodes instead of one. In other words, there exists no master node for an object. The scheme starts with a client starting a transaction at a local database server that consists of locking an object. This server now requests a lock on the object at all other nodes. If all other nodes granted the lock then a message is sent to the other nodes to perform the update and it makes sure that it is executed by all nodes using 2PC. The transaction is now committed to the client when the 2PC procedure was successful. This update scheme is called 'eager replication with group updates'.

The two update schemes are illustrated below [Wiesmann et al.], where the arrow in and out of the client represent the start and commit of its transaction and the gray arrows represent a time line:

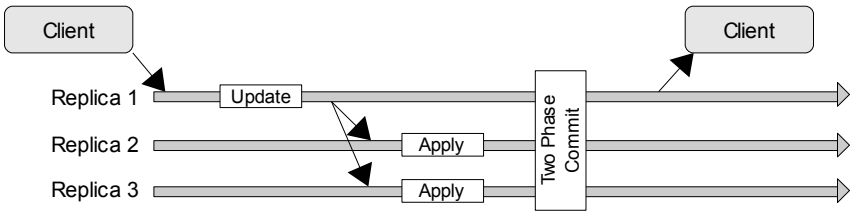


Figure 9.5 – eager replication with Master Updates using three nodes (replicas)

The master of the object is Replica 1 that first performs the update and then applies is at Replica 2 and 3. The 2PC block, that in fact includes the application of the update but drawn separate for simplicity, ensures that all node perform the update of not.

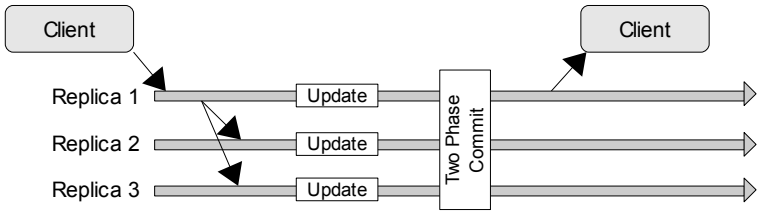


Figure 9.6 – eager replication with Group Updates using three nodes (replicas)

Replica 1 first initiates a lock on the object at Replica 2 and 3. Then the update is performed at all nodes what is made sure by 2PC. The main difference with master updates is that a client can also start a transaction at Replica 2 or 3 instead of only at Replica 1.

The locking steps within eager replication with group updates is until now considered as a lock at all nodes simultaneously. Some disadvantages of this technique are that no updates are possible when a node is unavailable (for a longer time) and that it involves communication with all nodes. There exist several alternatives [Silberschatz] that take away some of these disadvantages although they introduce other disadvantages:

- Single Lock Manager: instead of requesting a lock at all nodes, a lock is only required at one specific node, namely the Lock Manager. This implies that every database only has to have the lock granted by the Lock Manager instead of all nodes. A lock is only granted by the Lock Manager to one node at a time while 2PC still ensures that the update is performed at all nodes. This method saves communication and creates a less complex locking procedure and deadlock handling. Two disadvantages are that the Lock Manager can be a potential bottleneck and is a single point of failure.

- Multiple Lock Managers: this method is like the Single Lock Manager method, but now the objects are distributed over multiple lock managers. This approach decreases the chance of a possible bottleneck, but increases the chance of a lock manager being unavailable when all lock managers have an equal failure rate. Another disadvantage is the increase of deadlocks [Silberschatz].

- Majority Locking: instead of requesting and waiting for a granted lock from other nodes, this method reduces the number of nodes from which a lock needs to be granted. The number of nodes from which a lock needs to be granted is defined as at least half the number of nodes plus one. This implies that the advantage grows when the number of nodes increases. Another advantage is that it is possible to update objects as long as more than

50% of the nodes is available. Disadvantages are that it is quite complex compared to the previous methods and requires more communication. Also, deadlock handling involves a more complex algorithm.

- Biased Locking: this locking method distinguishes read-locks from write-locks. Read-locks can be granted easily by every node and only one read-lock is required. This creates the possibility for fast read actions from the database and doesn't require all nodes to be available. However, there is a drawback on the performance of writes, because an update requires a write-lock granted by all nodes just as the initial locking method of eager replication with group updates. This implies all nodes need to be available to perform an update.

9.4.2 Lazy replication

The main difference between eager and lazy replication is the moment when the client is informed about the commit or rollback of its transaction. Lazy replication uses only one replica to determine whether a transaction is committed instead of synchronization between all nodes. The steps of lazy replication are: starting a transaction at a node, update the object at that node and commit/rollback the transaction. After the commit at that node, the other nodes are updated.

Lazy replication has the same update schemes as eager replication, namely master and group updates. When the replication uses master updates then there is only one owner of the object that is used to update that object and this node always contains the most recent value of the object. After the update transaction at the master, it updates the values at the other nodes, the slaves. Using group updates there is no owner, or master, of an object. This means that an object can be updated at an arbitrary node instead of one specific master node. After the update transaction, the node updates the object at all other nodes.

The latter update scheme has an extra step, the reconciliation step, to ensure that all nodes agree on the value of the object because it is possible that two nodes update the same object at the same time. The conflict that can appear, is solved in this last step using a conflict solving algorithm such as described by [Greenwald et al]. The update schemes of lazy replication are illustrated below [Wiesmann et al.].

Q302

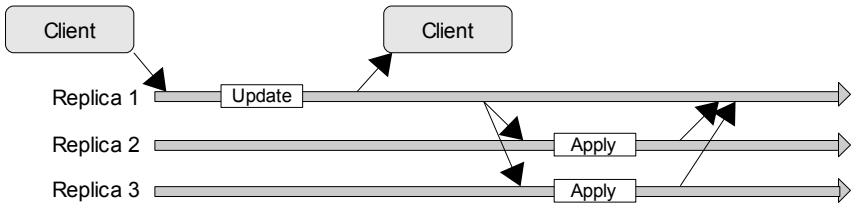


Figure 9.7 – lazy replication with Master Updates using three nodes (replicas)

Replica 1 is the master of the object and thus all updates on this object are done by Replica 1 what ensures serialization of the updates. It is however possible that the object is updated at Replica 1, but not yet propagated to Replica 2 and 3. Clients that read the object at those nodes read the old value of the object, i.e. a stale read.

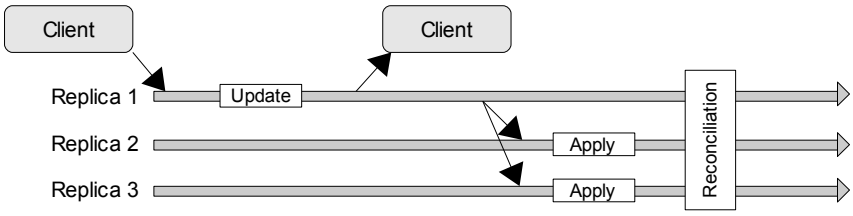


Figure 9.8 – lazy replication with Group Updates using three nodes (replicas)

The serialization that is guaranteed by performing all update transactions at one master in the previous figure, is unavailable when using group updates. It is now possible to update the object at Replica 1 and 3 simultaneously. This results in a conflict when both nodes are propagating their updates. A reconciliation block is used to decide which update will be accepted and which will be undone.

9.4.3 Two Tier replication

Requirement RQP002 introduced the requirement of nodes being unavailable. The two tier replication described by [Grey et al.] refers to these nodes as mobile nodes that suffer the same disadvantage of not being available all the time. When this requirement has to be met, eager replication doesn't supply a solution, because reads and writes both involve a transaction where all nodes need to be available.

Lazy replication allows nodes to be unavailable, but both of its update schemes introduce a specific disadvantage that also makes it less suitable to be used. The master update scheme has the disadvantage that a disconnected node cannot update an object that is not mastered by itself. The group update scheme allows this, but suffers from conflict resolving.

The two tier replication solution is based on both update schemes for lazy replication where nodes are categorized into mobile and base nodes. Base nodes are considered by be always connected and thus available while mobile nodes are disconnected most of the time. Equal to lazy replication with master updates, an object is mastered at exactly one node that can also be a mobile node but is usually a base node.

A replicated object at a mobile node can be a master or a tentative version. A master version is one that is equal to that of the master while a tentative version is an object updated locally at the mobile node but not yet at the master node. There are also two kinds of transaction, namely base and tentative transactions. Base transactions are transactions at those that work on a master object and involve one mobile node at maximum. Tentative transactions are performed on local tentative data and have to be redone later at the base nodes.

Several kinds of situations are now considered to illustrate the versions and transactions. If a base node would like to perform an update on an object that is mastered by itself or by another base node then the update scheme is equal to that of lazy replication with master updates. Since most objects are mastered by base nodes, this provides serialization without conflict resolving and versions will therefore usually be master versions. Q302

The main difference between two tier and lazy replication with master updates is that now a node is able to perform an update while not being able to connect to the master node, what is mentioned as a disadvantage of the that update scheme. In this case an object is updated locally using a tentative transaction resulting in a tentative version of this object. The term tentative refers to the fact that conclusions made from the value of the object can be undone. A mobile node has to connect to a base node to redo its tentative transactions and to update its objects to the master Q304

version. When a mobile node is connected to the network, it first sends updates of objects that are mastered by it. Next, it sends all its tentative transactions and accepts updates from the base node. The base node is able to make a conclusion whether a tentative transaction can be performed successfully or not and informs the mobile node about this. For the base node to be able to make these conclusions there is a rule called the scope rule [Grey et al.]: a tentative transaction may only involve objects mastered at a base node or at the mobile node that originated the tentative transaction. This rule implies that when a mobile node connects to a base node, the base node is able to communicate with the node mastering the object and thus make a conclusion that is reported back to the mobile node. If two objects involved in one tentative transaction are mastered at two different mobile nodes then this cannot be guaranteed, because the second mobile node can be unavailable, leaving the transaction tentative and thus unable to decide whether it can be committed or has to be rolled back. A tentative transaction, that failed during the execution of it by the base node, is considered as a problem of the mobile node that can decide to optionally redo it on the master version of the object or reconcile it with the user that knows it is using tentative transactions while disconnected.

9.4.4 General aspects of replication techniques

Before being able to compare the properties, advantages and disadvantages of the replication techniques of the previous, three general aspects of the replication techniques are considered. These aspects are scalability and blocking, availability of nodes and complexity.

Scalability and blocking

Several factors play a role in scalability and blocking, where a scale-up introduces more blocking. Eager replication doesn't only block a local database, but also all the others. When the number of nodes and operations increase as well as the number of blocks then the deadlock rate grows dramatically, namely as a third power of the number of nodes and the fifth power of the number of operations [Grey et al.].

Lazy replication with group updates doesn't suffer these deadlock rates, because locks aren't made system wide. Instead of a growing number of deadlocks, this replication kind suffers from a growing reconciliation rate. The reconciliation rate grows, just like the deadlock rate with eager replication, by a third power. This rate increases even more when mobile nodes are part of the network.

Lazy replication with master updates has the advantage of having a lock at only one (master) node, but is not suitable when mobile nodes exist. The deadlock rate for this kind of replication is quadratically, what is lower than that of eager replication.

Two tier replication has base nodes that use lazy replication with master updates and therefore have a deadlock rate that is also quadratically. Unfortunately, this kind of replication also suffers from reconciliation, because it is suitable for mobiles nodes that use tentative transactions.

Availability of nodes, connectivity

In a distributed database system a node can be unavailable due to many reasons such as power failure or lost connectivity. When a distributed database system needs all nodes to work then availability is an important issue. This is the case when using eager replication, but there exist several techniques to keep an eager distributed database system limited available without all nodes being available. Lazy and two tier replication are less reliant on nodes being available while still having a workable situation, especially with two tier replication, that is designed to have mobile nodes with poor connectivity.

Complexity

The replication alternatives have different complexity rates when it comes to transaction management, propagating difficulties such as reconciliation and preventing or solving deadlocks

[Grey et al]. Eager replication has complex transaction management, but doesn't suffer reconciliation. Lazy replication has less complex transaction management, but suffers from complex reconciliation when using group updates. Lazy replication with master updates is the least complex alternative, because it uses local transactions at the master and doesn't suffer from reconciliation. Two tier replication has complex tentative transactions that involve reconciliation, but this only applies for objects updated by mobile nodes.

9.4.5 Replication techniques compared

The previous paragraphs described several alternatives for replicated distributed databases and the different characteristics of their update schemes. In order to make a decision on which replicated database alternative presents the best solution for ELP and the two groups of network topologies, a comparison has to be made. This paragraph focuses on this comparison in three ways, namely:

- Which requirements of layer 1 (RQD001 to RQD008) are met?
- What specific advantages/disadvantages do the alternatives have for the first group that follows the order scheme and for the second group that doesn't?
- What practical remarks can be made?

First, the requirements are put together into one table to create an overview:

Property	Eager Mst. Upd.	Eager Gr. Upd.	Lazy Mst. Upd.	Lazy Gr. Upd.	Two Tier
Data integrity (RQD001-RQD004)					
Atomicity*	✔	✔	✔	✔	✔
Consistency*	✔	✔	x	x	x
Isolation*	✔	✔	✔	x	x
Durability*	✔	✔	✔	✔	✔
Distributed data (RQD005-008)					
Support for nodes being unavailable / Availability*	x**	x**	▪▪***	▪▪▪	▪▪▪
Any node can update the data*	✔	✔	✔	✔	✔
Suitable for non high-speed networks*	▪	▪	▪▪▪	▪▪****	▪▪****
Simple propagation of updates / Complexity*	▪	▪	▪▪▪	▪▪	▪▪
Scalability*	▪	▪	▪▪	▪	▪▪

Q301
Q503

* x - *Absent;* ✔ *- Present;* ▪ *- Poor;* ▪▪ *- Mediocre;* ▪▪▪ *- Good*
** *Biased Locking creates a limited read-only system that is not considered acceptable*
*** *Assumed that different objects are mastered by different nodes*
**** *Reconciliation consumes bandwidth*

Table 9.1 – properties of alternative replications techniques

Earlier paragraphs described that communication networks enable nodes to send each other progress messages. These messages can be used for one of the replication techniques. The best suitable replication techniques for each group can be given by combining the properties of the group with the properties of the replication techniques.

Group 1: following the order scheme
One important aspect of this group is that a node only 'shares' data with nodes in its parent-child relationship what will usually result in a replicated database with two nodes. This is always true for TransportableTrack/OutsourceTrack combinations. However, it is possible that a parent has more than one child and both children are not aware of each other. Assuming that the parent would like that the data of the transportable is equal at all involved nodes, i.e. itself and all its children, then this assumption rules out both eager replication techniques as well as lazy replication with group updates. The reason for this is that these replication techniques consist of communication between all nodes that isn't possible, because two children of the same parent are not aware of each other.

The only possibilities left are lazy replication with master updates and two tier replication. The latter is based on the first one with additional techniques, such as tentative transactions, to enable mobile nodes to be part of the replicated database. Both techniques require one master for each object in the distributed database. The parent is the only node that all involved nodes are aware of and thus this node has to be the master of the objects. This implies that if a child would like to update an object, it starts a transaction at the parent. After committing, the parent updates the object at all children. This supports the business situation in which an outsourcing company keeps control over what is executed. Considering the properties of table 9.1, the choice between lazy replication with master updates or two tier replication depends mainly on the presence of nodes with limited connectivity (mobile nodes). The decision for the best suitable replication technique is made in favor of two tier replication due to requirement RQD005.

A remark can be made about the TransportableTrack/OutsourceTrack combinations in which always exactly two nodes are part of the replicated distributed database. This wouldn't rule out the eager replication techniques, but the possibility of nodes being unavailable also leads to the decision to use two tier replication.

It has to be pointed out that this approach is conform the local-commits of paragraph 9.1.2. It is also possible to use two tier replication for tree-commit. This results in a string of nodes starting transactions at their parent up to the root. If the root commits the transaction then so do the children in the string down to the node that originated the update. If all nodes are base nodes, this can work well, but if one of the nodes is a mobile node then a tentative transaction exists, leaving nodes in uncertainty. Since group one primarily uses TransportableTrack/OutsourceTrack pairs to inform other nodes about updates, the tree-commit will rarely be used and therefore local-commits is assumed to be acceptable (TransportableTrack/OutsourceTrack pairs are only local-commit).

Group 2: not following the order scheme
The most important aspects of the second group are that the data of the tranportable is 'shared' among all nodes and that they are all able communicate with each other. This means that, in comparison to the first group, none of the replication techniques are ruled out for communication reasons. However, nodes being unavailable lead to the same two replication techniques being suitable for this group, namely lazy replication with master updates and two tier replication where two tier has the better support for this property. The decision for the best suitable replication technique is therefore also made in favor of two tier replication.

Choosing two tier replication for group two leaves one question, namely, which node is the master? Because every network topology and order tree starts with a root node this implies that this node will be the master at that point. This node also carries the greatest responsibility for being accountable and should therefore also be the one with the most accurate (master) data. The conclusion is that the root is the master node when using two tier replication in group two.

Practical remark
The major difference between the two candidates that use lazy replication, namely two tier and master updates, is that two tier is the better choice when mobile nodes exist. In contract, master updates is the better choice when they don't, because this replication technique supports isolation.

However, the advantage of isolation is not very big due to how updates are performed in practice: in practice there will usually be one transport company at a time that executes a part of the transport. This implies that situations where more than one node changes an object simultaneously are not likely to exist. Therefore the advantage of isolation is also not likely to be worth it while excluding mobile nodes from being part of the network.

Another practical remark that has to be made is the scalability property of each replication technique (RQD009). All replication techniques have a poor or mediocre score on this property. This implies that none of the techniques is suitable for situations with many nodes. Fortunately, two tier replication, that is considered the best replication technique for both groups, has a mediocre score on this property. Whether the scalability can become a problem depends heavily on the environment, such as resources (hardware, communication lines), and the reconciliation rate. Running a simulation that matches day-to-day usage can indicate whether the mediocre score becomes a problem.

Best suitable replication technique in case of high availability
Two tier replication is concluded as best solution for both groups. The reason to choose this replication technique is mainly because of requirement RQD005. If it is assumed that all nodes have a high availability then the best solution can be different.

The arguments that led to the conclusion of the first group indicated that most of the replication takes part between two nodes. As it is now assumed that these two nodes have a high availability, the eager replication techniques would definitely be candidates for a solution, because the network never expands to more than two nodes what prevents a large amount of deadlocks. The only property that can prevent this solution from being the most suitable is the bandwidth and latency of the communication network.

Eager replication is no candidate as a solution for the second group, because there is no limit on the amount of nodes being part of the network and, as described earlier, the deadlock rate grows by a third power of this number. Since lazy replication with master updates provides, just like eager replication, isolation, this would be most suitable solution for group two if all nodes have a high availability.

9.5 Layer 2: Dynamic replication participants and rights management

The previous paragraph provides solutions for the first layer that provides a working distributed database with a static number of nodes and no additional facilities. The second layer can be put on top of this layer to provide additional facilities that are mentioned by requirements RQD009 to RQD011. This paragraph therefore focuses on these three requirements:

- Real-time addition and removal of nodes (RQD009)
- Update rights management (RQD010)
- Support for removing data from the replication (RQD011)

9.5.1 Real-time addition and removal of nodes

A transport company outsourcing (a part of) an order adds a node to the order tree. This also involves adding this node to the network topology and the replication. The building steps in addition of the new node to the network are described in an earlier paragraph, leaving the addition to the replication left (RQD009). The addition of a node to a distributed network using replication can be described as adding a new node to the replication of an object. Since the best suitable replication technique of both groups is two tier replication, there exists one node mastering the object. The following situation describes the addition of new node:

The master of object X is master M. Node A is part of the replication and would like to add node B

to it. For the addition to be successful, two assumptions are made, namely that M is willing to add node B and that node B is willing to replicate an object mastered by M. The addition of B has to be performed in real-time implying that it is added during the normal operation of M and thus interfering with the transactions management at M. The result of the addition is that node B is part of the replication and has an accurate value of object X. The value of X has to be accurate, because M has to know from which point in time (version of X) it has to apply updates at node B (X is the most accurate version). The only way of assuring that the value of B is accurate is by examining it during a transaction at M. Initializing object X at node B requires that B is actually added to the replication (in reflection to the assumptions made). The conclusion is that either B has become part of the replication and M knows that at that point B has the most accurate value of X -or- non of these are true. A clear method of assuring this is by adding node B during a (special kind) of transaction at M.

The addition of B to the replication involves three nodes, namely A, B and M, that all require the addition to be committed or not. The well-known Two Phase Commit (2PC) protocol can be used to be sure whether the addition has been successful instead of being sure that an update has been successful.

Let T be a transaction that is initiated by node A containing the request of node B becoming part of the replication. During this transaction, master M as well as node B, write the same version of object X to stable storage including a record at M that node B has this specific version of X. This ensures that when 2PC succeeds, master M can continue to operate as it would when B was already part of the replication. In other words, M knows the version of object X at B and thus when it has to apply updates at B. The addition of node B using 2PC is illustrated in detail in appendix H that also includes a graphical representation of the 2PC steps performed by all involved nodes.

The removal of nodes can be done using an analogue algorithm. In fact, when this algorithm doesn't contain of any step to actually remove the replicated data, but only the fact that it is replicated, it presents a solution for the requirement to support the removal of data from the replication, mentioned by RQD011.

9.5.2 Rights management

The last requirement of paragraph 9.3 that is not yet taken in consideration is update rights management (RQD010). This requirement prescribed that it has to be possible to grant and revoke rights of one or more nodes to update data that is 'shared' by these nodes. This implied that there has to some kind of hierarchy between those nodes. This part of the chapter focuses on this last requirement by considering three alternatives that can be used for granting and revoking rights.

Q006
Q404
Q508

Before focusing on the possibilities for implementing a rights management technique, the hierarchy of the two groups is examined. The first group consists of many small hierarchies between every child and its parent. The child here cares the responsibility and if this child decides to also outsource its order then this is of no interest for the parent. This is also supported by the data of a transportable not being 'shared' between all nodes. From this it can be concluded that the first group has no real reason to have a grant/revoke rights management technique, because if a node outsources its order, it would definitely grant rights to the child since it wouldn't receive its progress information otherwise. A practical remark that can be made on this, is that it is assumed that a company only has power over its own business and no influence on the company it outsourced its order to, see paragraph 6.5. Another (trivial) agreement can be that a company, that is not the root nor a leaf in the order tree, is obliged to update its shared data in such a way that progress message are propagated to all companies involved.

In contrast to the first group, the nodes in the second group all share data of the same transportable. This means that every node can update that update that data, even if it is already

finished with its part of the transport for a long time. The possible agreements between companies as described in paragraph 6.5, can contain an agreement that only the root company in the order tree is allowed to change an order after it has finished or change the original location and final destination. This means that there is a good reason for a rights management technique in the second group. This part is therefore only relevant for the second group.

Granting and revoking

Q403

Paragraphs 8.1 briefly describes the creation of the order trees of use case 3 and 4. An order tree describes what company outsourced (a part of) an order to another company and always starts with a company in the root that received the order from the client. As mentioned in the goals of the use cases in paragraph 5.1, outsourcing of an order also handles over the responsibility of the order. When multiple companies are involved in executing the order then they will all update the data of the transportable for the progress that is made by them. However, since only one company is really executing the transport at a time so there is no need for all the companies in the order tree to be able to change the data of the transportable at the same time. The company that is executing the order also doesn't like this, because this company caries the responsibility of the goods at that time and doesn't like interference by other companies that are changing the data. In short it can be said that when a company hasn't got any reason to change the data then it should not be able to do so.

To create a rights management solution there has to be focus on the companies that carry responsibility at a moment. This is illustrated using the figure below.

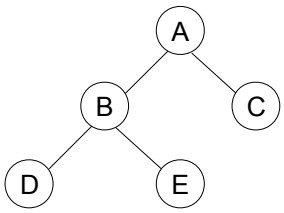


Figure 9.9 – initial order tree without rights management

First, only company A has the responsibility of the order. This company is also the one that always carries the responsibility of the order to the client, even when it is outsourced. During the next steps the order is outsourced and an order tree as displayed above can be drawn. In this case, company D is responsible for the transport of the goods from the original location to the warehouse of company E. Next, E has to transport the goods to the warehouse of company C and company has to deliver the goods at the final destination. When company D is executing its transport then there is no need for company E and C to change the data of the transportable. To provide this situation, A grants update-rights to B followed by B granting these rights to D. Altogether there is a "grant track" from A to D as displayed by the thick blue line in figure 9.10 below on the left.

Figure 9.10.a-b – order tree with a grant track from A to D (9.10.a); order tree with a grant track from A to E (9.10.b)

After delivering the goods at the warehouse of company E, company D updates the data of the transportable and its job has finished. This implies that D doesn't need its update-rights anymore. Therefore they should be revoked by the company that granted them, i.e. company B. This company now grants these rights to company E what results in the order tree in figure 9.10.b above. The companies that are able to update the data are now A, B and E. Although the other companies are not able to update the data, the do receive updates of it. The next steps would be that B revokes the rights from E, A revokes the rights from B and grants them to C.

The following rules are used to grant and revoke update-rights from and and to other companies using a tree structure with companies as nodes:

1. The root node always has update-rights
2. Any node can grant rights to its children
3. Any node can revoke rights from its children
4. When a nodes' rights are revoked the so are the rights of its children (it it has any)

It is not prohibited to have a situation where all nodes have update-rights. Also, it is not prohibited to revoke rights from a node that is going to or is already executing (a part of) the transport, but this would not be a suitable situation.

As an addition to update-rights there can also be outsource-rights that define whether a node is allowed to outsource its order. The appliance of these outsource-rights can be analogue to the update-rights.

Rights administration

Using the rights management solution, described in the previous paragraph, it is possible to have update-rights granted to exactly those companies that are responsible for the execution at a certain point in time due to the grant track in the order tree. To apply the rights management solution there has to be a guard that allows or rejects updates of the data. There are several alternatives to introduce a guard which are discussed in the next paragraphs. The three alternatives are:

- External single guard
- Internal guard group
- Internal single guard

External single guard

When this alternative is used then there exists a single external party, i.e. a party that is not represented by a node in the order tree, to control the update-rights. This guard must be initialized by the root node to create a record of the data together with the trivial initial rights. When a node is added, initially by the root node, to the order tree then this is registered at the guard together with the rights that are granted to it by the root node. The next step can be that the added node is also adding another node, for example company D in figure 9.10.a. This can only be done if any added child (by the root node) is aware of the guard to register new nodes and granted rights. In short it can be said that when a node is added to the order tree then it has to be provided with the information about the existence of the external guard and every added node is registered at it. This results in a guard knowing all the nodes, their relation to each other and the rights that are granted. The guard is also able to perform the the last rule that is mentioned in the previous paragraph. When a node would like to update the data then the guard can easily decide whether the update is allowed or not. This decision of the guard can easily be requested by the root mastering the data and actually performing the update.

Internal guard group
This alternative uses the nodes of the order tree to control the rights management. This is done by negotiation of the nodes that are part of a grant track. When a node, for example company B in figure 9.10.a, would like to grant update-rights to a new node, company D, then it requests this from the members of the grant track, i.e. in this case only company A. If company A accepts this grant-request then company D is able to perform updates and vice versa if A rejected the request. When there are more companies in the grant track then there are several alternatives for the decision making. One could be that all nodes have to accept and another can be that the majority has to accept the grant-request. A revoke of update-rights is not negotiated and is accepted immediately to prevent a node being aggravating.

Internal single guard
This alternative is analogue to the previously mentioned *External single guard* alternative. Instead of having an external party that has to function as a guard, a node of the order tree performs this role. The node that performs this role has to be the root node, because this is the first node that can add another node and defines the rights of this new node. Together with the rules for granting and revoking this implies that it is the only node that is always present in a grant track.

Advantages and disadvantage
The three described alternatives will all be able to administrate the rights that are granted and revoked within the order tree. To choose the best alternative, the advantages and disadvantages of them need to be compared.

The *External single guard* has an advantage that the guard has a dedicated role that has no interest in the context of the order tree and the execution of the order (unbiased). Another advantage is that this alternative is not complex. A disadvantages is that it is an extra node that has to be able to communicate (and understand) with all the nodes in the order tree. It is not acceptable to be unavailable.

The *Internal group guard* has one big advantage over the other alternatives which is that it introduces a democracy for granting and revoking rights. However, this is not of any use because rule number two defines that every node can grant rights to its children as long as it has these rights itself. A disadvantage of this alternative is that the democratic process adds (unneeded) complexity. Another disadvantage is that every node in the grant track has to be available for a leaf to grant rights.

The last alternative, *Internal single guard*, has the advantage that it is not an extra party that introduces the extra risk of a party being unavailable as in the first alternative. Other advantages are that this alternative is not complex and that the root node, that has the responsibility to the client, has a complete view on the grant tracks. A disadvantage of this alternative is that it can cheat, because the guard has an interest in the context and is the only node that keeps track of the rights. However, it would not need to cheat, because the root node is already able to revoke update-rights from all the other nodes in the order tree.

The recommended alternative would be *Internal single guard*, as it is not complex and in agreement with the rules for granting and revoking update-rights. The root node also is the master when two tier or master updates replication is used, resulting in one node that controls and performs the updates what is in line with it carrying the final responsibility to client.

9.6 Summary and final design decisions

This chapter is the last chapter that focused on the design of ELP. Therefore it is first summarized followed by final design decisions based on the chapter 7 through 9 that focused on technological aspects to create an information system that implements the business processes of chapter 6.

9.6.1 Summary

Using the CDM it is possible to describe the contents of an order for a company that outsources an order to one or more other companies. All these companies have their own part of the transport track that they have to execute using the TransportTrack entity. If a company outsources an order then it has additional information described by the OutsourceTrack entity to know which part of the transport track is outsourced to which company. This OutsourceTrack information has an equality relation with the TransportTrack entity of the company that the order is outsourced to. The information that has to be equal for all involved companies, such as the goods specification and TrancportTrack/OutsourceTrack equalities, is kept up-to-date using a distributed database technology.

During the execution of a transport a delay can occur. Companies that are downstream in the flow of goods scheme can be informed about this by either changing the TransportableTrack or Transportable information, respectively depending on the network topology group. The latter suffers the disadvantage of publishing confidential information, although this was already the case for the second group.

To propagate changes of information between all companies, the information at all companies is seen as a distributed database with identical copies at two or more nodes. Changes to this information can be propagated using three replication techniques, namely eager replication, lazy replication and two-tier replication. Eager replication is not suitable for ELP because problems occur when communication channels are not available. Lazy replication and two-tier replication have better support for communication channels being unavailable. If mobile nodes exist within the distributed database then two-tier replication is the best option, while lazy replication (with 'master updates') is the better option when they don't.

Both two-tier and lazy replication have a single node that masters a piece of information. It is not always preferred if every node can update this information any time what can be limited by introducing a rights management technique. There are three alternative techniques, namely external single guard, internal group guard and internal single guard. Internal single guard is the recommended alternative, because it not complex and in line with responsibilities and both suitable replication techniques.

9.6.2 Final design decisions

Several of the design decisions of paragraph 4.2.7 have influenced the research and the results of chapter 7 to 9. Chapter 7 described the idea to introduce a Common Data Model (CDM) for transportation industry that can be used to represent information in a uniform way. This information is, usually, stored in a local database using a proprietary data model. The proprietary data model of the two software product of Global Data Exchange can be mapped to the CDM of chapter 7. This enables at least two Transport Management System products to exchange information. In general, two or more information systems can only exchange information if they both understand and support the same way of describing information that is required to conduct electronic business. This is exactly what the CDM provides.

One of the specific design decisions for ELP related to the CDM is to support entities that are required to (partially) outsource orders as well as to be able to Track and Trace them. Using the TransHolder entity of the CDM the latter is even possible when goods are contained into other goods such as sea containers. This part of the CDM design is innovative although participants have to agree on set of rules (appendix F) and it has to emphasized that the success of it can be influenced by the fact that outsourcing and exchanging information about progress crosses company borders, see paragraph 6.5.

Although it is not certain whether all existing TMS product are able to map their proprietary data model to the CDM, the CDM is considered as a good starting point to provide a base to describe

entities that exist in TMS products. Chapter 9 shows that the design of the CDM provides all the required entities and attributes to exchange progress information if outsourcing is involved, enabling users of ELP to be held accountable and to provide real-time Track & Trace information.

One of the properties of the CDM design that is not used in chapter 8 or 9 is the possibility to extend it with user-specific attribute requirements and extensions. Although the design decision cause 12 in paragraph 4.2.7 described this as a design decision that increases flexibility and possible acceptance, there exists no sign of it to be required. Altogether, the design decisions that influence the design of the CDM are not all equally relevant (11, 13 and 14 show their relevance while design decision 12 doesn't).

Chapter 8 focuses on possible network topologies that can be used to create communication networks as well as alternatives to send progress messages depending on network topology properties. The main aspect of this chapter is that it introduces two categories with a different communication strategy. The first category consists only of one-to-one communication between an order outsourcing/accepting pair. The second category consist of one-to-many communication between almost all involved participants. These two different groups created an unforeseen split in solutions that was not thought of when the design decisions were made. In fact, the design decision whether the first or the second group communication strategy provides a better solution is still open, because only the two alternatives are given with their advantages and disadvantages. In general, the first category stays closer to practice while the second category provides a better (no conflict, faster, more reliable, less complex) technological solution. A great technological solution can be worthless if it is unacceptable for the participants from a business point of view, creating a slight but yet unfounded preference for the first category.

One design decision of chapter 9 solved many problems, especially on the subject of conflict. This design decision is to have one master nodes that ensures serialization of updates. In fact, the tweaks to the local-commit in paragraph 9.1.2 and the suggestion to use local-commits up to the root in paragraph 9.4.5 illustrates that lazy replication with master updates can be considered the best distributed database design for ELP if no mobile base nodes exist. If they do, then two-tier, what can be considered an altered version of lazy replication with master updates, is considered the best design decision between the available replication techniques. Two-tier however, disables the 'local-commits up to the root' idea of paragraph 9.4.5 if a node within the chain is a mobile node. This is why lazy replication with master updates including the 'local-commits up to the root' idea suites the first category of communication best. If the second communication category is preferred then this addition is not necessary.

At the end chapter 9, design decision 10 is taken into consideration. The given alternatives provide ways to control the ability of participants to make changes to data that is of interest to many participants. This design decision and the presented alternatives are considered a valuable addition, especially because it can exist on top of the suitable replication techniques (lazy replication with master updates and two-tier) and is in line with the responsibilities.

10 ELP Prototype

The previous chapters described the existing solutions, design decisions, standard business processes, Common Data Model, exchange of information, communication solutions, replication techniques for synchronization and rights management. All these designs and solutions can be used to develop an ELP prototype. It is not possible to create a full implementation of ELP, because several essential parts, such as full standard business process specifications, are not available.

The ELP prototype can especially support those parts that have been described in more technical detail. These parts are the CDM, communication solutions and replication techniques. Naturally, a prototype that implements these parts should take the design decisions in chapter 4 in consideration. This chapter focuses on the development of the ELP prototype and includes goals that one would like to achieve, the prototype architecture and implementation details. The next paragraph first focuses on the goals that one would like to be achieve.

10.1 Goals of the ELP prototype

Before and after implementing a prototype, several questions can be asked about it, namely [Borysowich]:

A) What is actually prototyped?
B) What type of prototype is going to be made?
C) What can be learned from the implementation of the prototype that can be used in a real implementation?
D) Which design decisions can easily be adopted and which cause problems in a real implementation?
E) Is it possible to measure performance using the prototype as a simulation?

First, questions A and B are answered in this paragraph. Next the design and implementation details of the ELP prototype are given followed by the answers to questions C, D and E. There exist a number of prototype types [Borysowich] that all have a typical purpose. The purpose of the ELP prototype is to test the key functions of the event-based business process 'Provide Status Information', because this is one of the design decisions of ELP that distinguishes it from existing solutions and chapters 7, 8 and 9 provide detailed information that can be used. The type of prototype is therefore a Vertical Prototype that has the general characteristics of demonstrating a working, but incomplete, system for key functions. The prototype uses the contents and decisions of chapters 7 through 9 to see whether these are suitable to create a successfully working prototype.

The scope of the ELP prototype is limited to the key functionality of the addition of a new node to create or extend a replicative situation and the ability to update a replicated object by every node. An instance of the entities of chapter 7 can be used as object, for example a client. To define more precisely what will be part of the prototype it is possible to divide the design decisions into two categories, namely those that are taken into consideration and those that aren't.

Design decisions used within the prototype

Design decisions 3, 4 and 5
These three design decisions consist of ELP using messages between participants that are formatted using XML and can be transmitted using several transport methods, such as HTTP and SMTP.

Design decisions 6

The ELP prototype uses the ELP Name Service as a registry to look up information that is needed to communicate with a node identified by its ELP Identifier.

Design decisions 7 and 1
The ELP prototype is only used for the specific part of 'Provide information status' where changes to an object, that contains 'current location information', are sent to other nodes.

Design decisions 9 and 13
All the participants that use the ELP prototype need to have accurate information. This is achieved by using a replication technique that propagates updates and that allows determination of the most accurate value of an object. The number of nodes that replicate an object can be increased in real time.

Design decisions not used within the prototype

Design decisions 1, 2, 8, 10, 11, 12, 14 and 15
The ELP prototype only supports a part of the business process to provide status information and therefore none of the other business processes as well as specifications of custom business processes. Neither does it support the extension of the data model and transport up-scaling. The ELP functions that are described by these design decisions all require that most of the other design decisions have been implemented. Since these design decisions would only be refinements and additions to the key functions they will be not part of the prototype.

10.2 ELP Prototype Architecture

Before the key functionality can be implemented in the prototype it is required to create an architecture that consists of all the components that will be part of the implementation. There are many requirements and design decisions that would lead to a complex architecture. Although not every requirement and design decision will be part of the ELP prototype, it still consists of many parts, such as communication interfaces, data storage and business logic.

A design pattern that is available for complex applications is the Model-View-Controller design pattern that has been designed by [Reenskaug] in 1979 and adopted by many international organizations, for example Sun Microsystems [MVC-pattern]. This pattern makes a complex application more manageable and improves maintainability and extendability. This is done by separating business and control logic from the data presentation. The application is hereby divided into three layers, namely the model, view and controller layer. Although originally designed for object oriented Smalltalk applications with a user interface, it is also very suitable for the ELP prototype. The three layers have the following characteristics, but are not very strict, leaving some flexibility to the designer. Naturally, it is possible that multiple parts of the application belong to one layer.

Model layer
The definition of the model layer given by [Reenskaug] is: *a Model is an active representation of an abstraction in the form of data in a computing system.*

The model layer manages data of the application. Therefore the primary property of the model layer is that it encapsulates the state of the application. The model in the ELP prototype typically manages the connection and queries to the database.

View layer
The definition of the view layer given by [Reenskaug] is: *to any given Model there is attached one or more Views, each View being capable of showing one or more pictorial representations of the Model on the screen and on hardcopy. A View is also able to perform such operations upon the*

Model that is reasonably associated with that View.

The view layer consists of the interface to the users of the application where it can present data in one or more forms. Changes in the model or controller don't necessarily imply changes in the interfaces. Although the documentation of Sun Microsystems about the MVC pattern focuses on web browsers that use the interfaces, the users of the interfaces of the application can also be automated systems. In the ELP prototype, the view layer takes care of the communication and formatting of messages.

Controller layer

The initial design of the MVC Pattern by [Reenskaug] did not consist of a Controller. Instead, an Editor was defined: *an Editor is an interface between a user and one or more views. It provides the user with a suitable command system, for example in the form of menus that may change dynamically according to the current context. It provides the Views with the necessary coordination and command messages.*

The Editor is replaced by a Controller and extended with a (new) more specific Editor that is not described in detail here [Reenskaug]. The controller layer is in fact the glue between the model layer and the view layer and defines the applications behavior. It takes care of executing business logic in response to a received requests of the view layer. The result of the business logic executed is afterwards passed to the view layer to create a reply to the earlier received message. Changes in the state are passed to the model layer to become persistent.

The three layers of the MVC design pattern clearly introduce separation of concerns. As long as the interfaces between the three layers remain the same, the three layers can be developed, changed and maintained independently by separate groups of developers that all have their own specialty.

Now that the common characteristics of the three layers have been given, it is possible to divide the design decisions into the three layers. Design decisions 3, 4, 5 consist of information about formatting messages and communication between nodes. This information is therefore part of the view layer that can represent multiple interfaces. This is especially useful for the decision to support multiple transport methods. Received messages of the view layer are passed to the controller layer that doesn't care how the messages have been received. Also, when a message needs to be sent to another node, design decision 6 describes that the ELP Name Service is used to find out which communication means are available and how that node can be reached. This is also not of any interest to the controller and thus the ELPNS functionality will be in the view layer.

The other design decisions that will be used in the prototype can be distinguished by whether they are involved in the administration and enforcement of object replication or that they are involved in business processes, such as making changes to an object to provide information to the other nodes. All of this functionality belongs to the controller layer although they will be in separate parts of it. Finally the model layer takes care of persistent state changes. The MVC design pattern is used to create the ELP prototype architecture that is illustrates in the following figure.

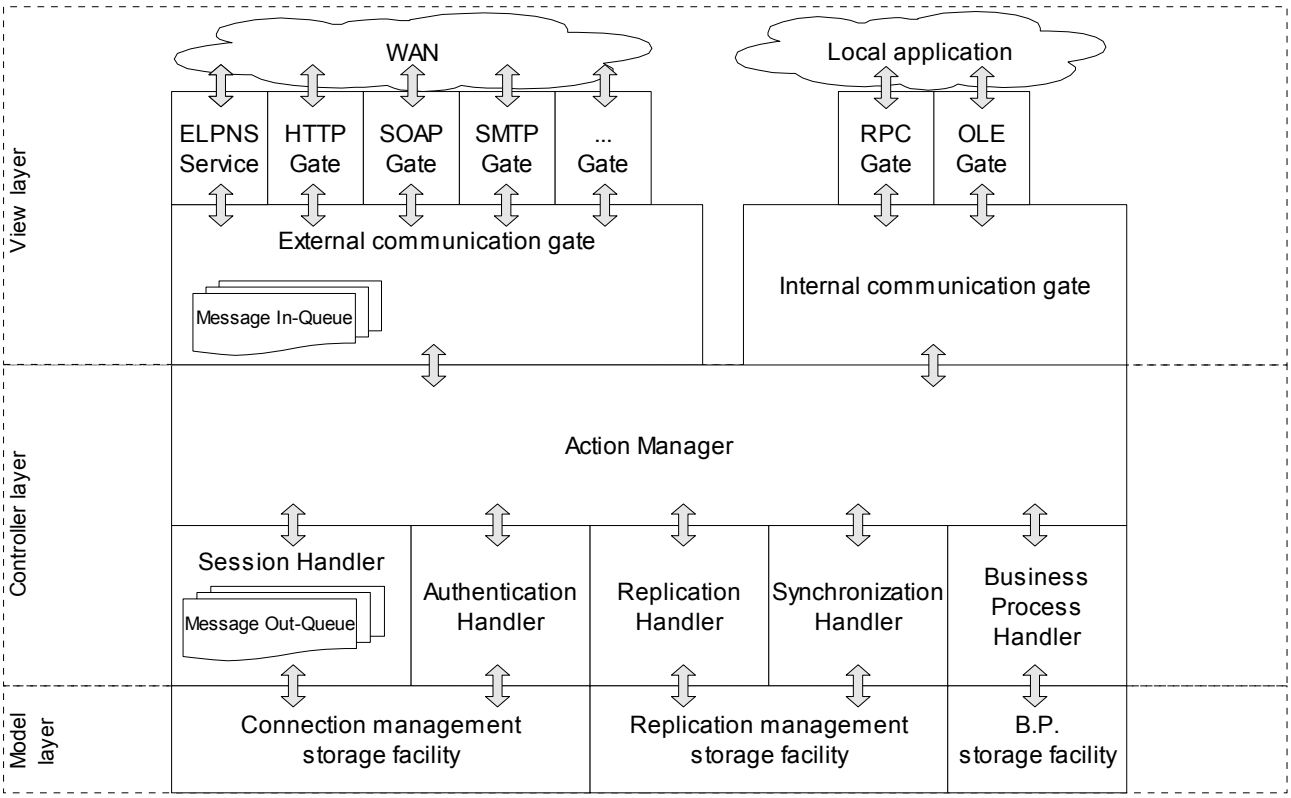


Figure 10.1 – ELP prototype architecture

The MVC design pattern has many similarities with the Three Tier Software Architecture [Sadoski]. One of the differences is that the MVC design pattern allows the view layer to use the model layer directly. Although the ELP prototype in its current form doesn't require this functionality, it can be useful for the final implementation. Two examples of the view layer using the model layer are the request of message templates from a storage facility and the use of an administration containing information about ELP identifier lookups for caching purposes.

The following three sections describe the functionality of the ELP prototype for every layer.

View Layer

The view layer of the ELP architecture consists of gates that take care of communication with external participants, for example other transport companies, and internal parts, for example the local information system of a transport company. In short, the view layer provides and internal and external communication interface.

The external communication gate provides at least one method (SendMessage) that enables the action manager to send a message to an external receiver. This method therefore requires two parameters, namely the address (ELP Identifier) and the message. Regardless of the transport that is going to be used to send the message, the messages sent by the action manager are always formatted identically. The role of the view layer is to alter the presented information (message) in such a way that it conforms the format that is expected and usable by its user, in this case the receiver. Although no reformatting might be required, it can be used to introduce functionality to be able to send messages according to, for example, RosettaNet message formats. Q106

After the reformatting by the external communication gate it can be required for a specific transport gate that the message needs to be reformatted again to comply with technical requirements of that gate. An example of the latter reformatting is done by the SMTP gate that has to encode the

Unicode XML message to Base64 due to the limited number of characters of the ASCII character set that can be used within e-mail messages, namely 127 characters. Receiving messages is also done by the transport specific gates and the external communication gate. Receiving a message has exactly the reversed formatting process as sending a message.

It has been mentioned that an ELP identifier is used to address a message. The advantage of this is that an ELP identifier is independent of the transport method used. This introduces separation of concerns between the view- and the controller layer about message addressing, because the controller layer can always use a single type of destination address regardless of the transport method used. If the SendMessage method of the external communication gate is called using the two arguments then it needs to lookup the ELP identifier from the ELP Name Service. The ELP Name Service provides the external communication gate the information about possible transport methods and requirements to use them. The ELP Name Service is described in more detail later on in this chapter.

The following figure illustrates the sending and reformatting of the external communication part of the view layer.

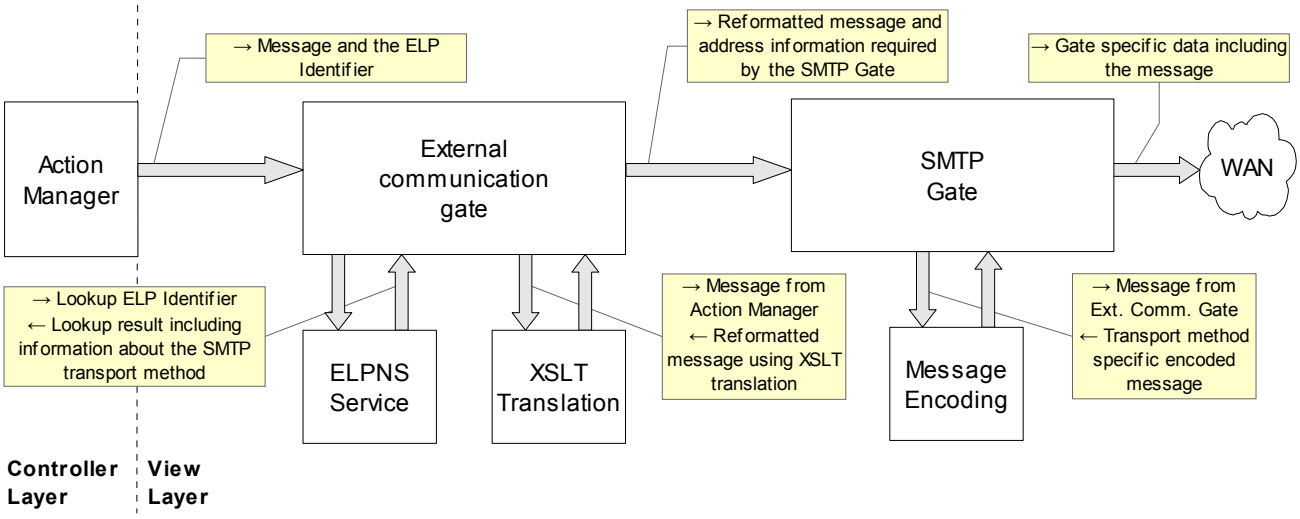


Figure 10.2 – sending a message using the external communication gate

If a message is received from one of the transport method specific gates then it ends up in the message in-queue of the external communication interface. After the message is enqueued, an event is called at the action manager to inform it about the presence of a received message. The action manager can handle this message immediately, but it is also possible that it postpones the handling of the message due to, for example, a high system load.

If a message can not be delivered to the intended receiver, this influences the flow of the current business process or business logic used for replication. In these cases the session handler is notified about this failure. The functionality of the session handler is described in more detail within the paragraph about the controller layer.

Another part of the view layer is the internal communication interface. This interface functions as anchor for existing applications to integrate ELP functionality. If ELP is used as middleware then this interface can consist of Remote Procedure Calls. Naturally, if ELP is integrated more deeply within an application then it can consist of interfaces that are more specific for the development and system environment.

◈ Q506

Controller layer

The controller layer consists of two sublayers, namely the Action Manager and its helpers. The action manager is the the central communication part that performs several important tasks:

- Creation and initialization of all surrounding handlers and interfaces
- Creation and initialization of data storage facilities used by ELP
- Transfer received messages from the external communication interface to the session handler
- Transfer messages that have to be sent to the external communication interface
- Trigger the resending of messages at the session handler
- Transfer received messages to the correct action handler (replication, synchronization and business process)
- Transfer messages that have to be sent from the action handlers to the session out-queue
- Transfer messages from the internal communication interface to the correct action handler and vice versa

An 'action' can be defined as communication between two or more nodes to implement a specific functionality. Examples of actions are “Update object X at node Y” and “Add node Y to the replication of object Z”.

In short, it can be said that the action manager functions as a coordinated intersection between all its surrounding handlers and interfaces. The surrounding handlers are referred to as action handlers.

Session Handler

The session handler takes care of the building of a session between two ELP nodes. Before the two nodes can communicate with each other in a functional way ('action'), they first have to set up a session. One node requests a session at the other node. This node can agree or disagree on the set up of the session. There exists a separation of message categories, namely those that are used to set up a session and those that are used within actions. The session handler knows which messages are used to set up a session and the sequential order of them. However, the action messages belong to functionality of one of the action handlers and the contents of them is out of the scope of the session handler. In short, the session handler handles session messages by initiating messages and sending replies, while it only functions as a gateway when it comes to action messages.

The session handler records all the last incoming, outgoing and to-send (out-queue) messages of the session. The action manager asks the session handler on a frequent basis for messages that have to be sent, although this can also be done using events. These messages are handed over to the external communication interface for further processing. An incoming message is stored for every session together with the result of the event that was executed due to the contents of the message.

The session handler takes care of retrying of already sent messages. The session handler stores the last sent message for every session. If there is no response from the other node within a certain amount of time then it resends the message. If a message is received twice by a node (a node knows its last received message) then it sends a notification that the message has been already received and that the retry attempts can stop. The session handler uses sequence numbers for every non-session (i.e. action) message so that it can distinguish old from new messages. If a message is delayed for a long time and a retry-message has already reached the node then it can use the sequence number to know that it has received an outdated message.

If the session handler has successfully set-up session(s), which are required by the action handlers, then it notifies the action manager which, in turn, notifies the correct action handler for the active action waiting for the session(s). If authentication is required then this handler will be the

authentication handler. If a session fails during the execution of a business process or replication action, for example due to time-outs, then the action handler is notified in an analogous way so that it can undo any temporary changes. Additionally, a session is automatically closed after a certain time of inactivity, regardless of whether an action has finished successfully or not.

The main goal of a session is to provide an authenticated and reliable communication channel that can be used by an action handler without them knowing anything about the underlying techniques used. On the other hand, the session handler has no knowledge about the functionality that is implemented using the sending and receiving of action messages.

Authentication Handler
The authentication handler is an action handler that takes care of the authentication of a session. If a session requires authentication then the authentication handler tries to authenticate the session using a password that is required for the other ELP node. If a session has been authenticated (or failed) then it notifies the action manager that it can start the action that was waiting for the authentication (or to abort it).

Replication Handler
The replication handler handles all messages that are involved in the building/destroying of replication structures. This handler can add slaves to a replication or can accept to be a slave in another replication. The following items are part of the replication handler functionality:

- Another master of a data structure would like the node to become a slave of a replication
- The node would like add another site to the replication
- The access control to data structures (assignment and revocation of write access)

Synchronization Handler
The synchronization handler handles all messages that are involved in the synchronization of a data object. The synchronization handler must have the ability to receive a message from the internal communication interface that tells this handler which data object is updated and should be propagated. This handler also receives messages from the external communication interface that can tell the synchronization handler to update certain data objects.

Business Process Handler
The business process handler takes care of the business processes supported by ELP. One of the mentioned business processes is to inform other ELP users about the progress of the execution of an order. To be able to this, it is assumed that there already exists replicated data that is used for this purpose, see chapter 9. This replication is set-up by the replication handler and the synchronization is controlled by the synchronization handler. To give a better total view on the, previously described, helpers (handler) of the controller, the following example shows which task is performed when the location of a transportable is changed.

First, the following basic assumption is made:
- Transportable X is a replicated object at nodes A and B where A is the master
- There doesn't exist any active session between the two nodes
- Authentication between the nodes is required and the authentication information is present
- Both nodes have an ELP identifier and a transport method that is supported by the other node
- All messages between the nodes arrive normally (no delays or loss of messages, etc)

The scope of this example is node B that would like to change the location of transportable X. This involves starting the business process to update the progress information. This business process is started by a request to initiate it which is received through the internal communication gate by the business process handler. This handler requests the update of the data of the transportable at the synchronization handler. This handler is aware of the other nodes that are present in the

replication; in this case only node A. It is emphasized that the business process handler simply requests the update and the replication of it is not of any concern of this handler (principle of separation of concerns).

The synchronization handler is aware of the fact that the data of transportable X is mastered by node A and that the lazy master replication with group updates requires that the update is performed at the master. To be able to this, a communication session has to be started between node A and B. This session takes care of several aspects that the synchronization handler takes for granted, such as the re-sending of messages if they are assumed to be lost. The synchronization handler requests a session with node A at the session handler. The session handler initiates this session with node A by sending a session request message to it. Node A replies to this request with a message that authentication is required. The session handler now requests the required authentication information for node A from the authentication handler. Assuming that this information is provided, the session is created successfully and now functions as a carrier for ELP messages.

The successful creation of the session is reported back to the synchronization handler that requested it. The synchronization handler uses the session handler to send messages to node A and received messages are passed through to the synchronization handler. This implies that the session handler has to be aware of which helper is using a session and that the helper is aware of which session it is using. For simplicity, within the prototype it is assumed that every helper only performs one operation simultaneously and that only one session exists between two nodes. The message types are simply used to forward received messages to the intended helper. After the synchronization handler has updated the data of transportable X at node A (and therefore also locally), it reports this result back to the business process handler that was still waiting for the operation to be completed. This results in a successfully executed business process.

At this point the synchronization handler and the business process handler both respectively finished their update operation and businesses process. The only thing left is the session between node A and B. This session will be automatically closed by one of the session handlers after a period of inactivity. This functionality saves the overhead of creating a new (authenticated) session for every business process that frequently involve the same nodes.

Model layer
The model layer of the ELP prototype only consists of storage facilities for the controller layer to request and store data that is used within the helpers of the controller layer. These storage facilities typically consist of connections to database systems. The advantage of the model layer is that the controller layer doesn't need to be aware of any database system specific properties such as SQL dialects. Although the architecture presumes that several database systems are used, the ELP prototype only has one database system, because this suffices the narrow scope of the ELP prototype.

10.3 ELP Name Service

The previous paragraph mentioned the ELP Name Service (ELPNS) that is described by design decision 6. The communication with this service is done by a special part of the view layer that provides ELP identifier lookup functionality and is put on the same level as the HTTP, SOAP and SMTP gates. The reason for this is that the result of an ELP identifier lookup defines which gate(s) can be used by the external communication gate. This paragraph describes the ELP Name Service in more detail.

The ELP Name Service is a service that provides a translation from an ELP identifier to one or more possible transport methods and their required information that can be used to communicate with the node identified by the identifier. The ELP Name Service is a centralized service that

functions as a central registry of ELP users and their possibilities to communicate with each other. If a node would like to communicate with another node then it looks up the communication information at the ELP Name Service and chooses one of the possible communication methods. An ELP identifier lookup consists of the following two steps:

1. A node sends a lookup request to the ELP Name Service
2. The ELP Name Service replies with a collection of transport methods and the required information to be able to use each of these methods

A typical example of the ELP identifier is “gdx”. A lookup result of this identifier can, for example, consist of two transport methods, namely a STMP transport method with an e-mail address as required information and a HTTP transport method with a URL, that has http as scheme, and POST as request method. Although a single URI or URL can initially be considered sufficient for the two transport methods it has the disadvantage that these descriptors cannot always provide all the required information such as the HTTP request method.

Sending and receiving the ELP identifier and its lookup information can also be done using multiple transport methods, although, because of its centralized characteristics, these are considered static and comparable to the Internet Root Name Servers [RootDNS].

Despite that the example of the ELP Name Service only provides information that is meant to be used for ELP purposes, it is possible to extend the service and the example in such a way that it provides a more generic lookup service that can better suite future requirements and thus extension of the protocol. This additional functionality is easily added by introducing two extra lookup parameters, in addition to the ELP identifier, that describe the service and version of which information is requested.

ELP Name Service Architecture and design

The architecture of the ELP Name Service has many similarities with the ELP prototype architecture. It is also based on the MVC pattern, but has fewer components.

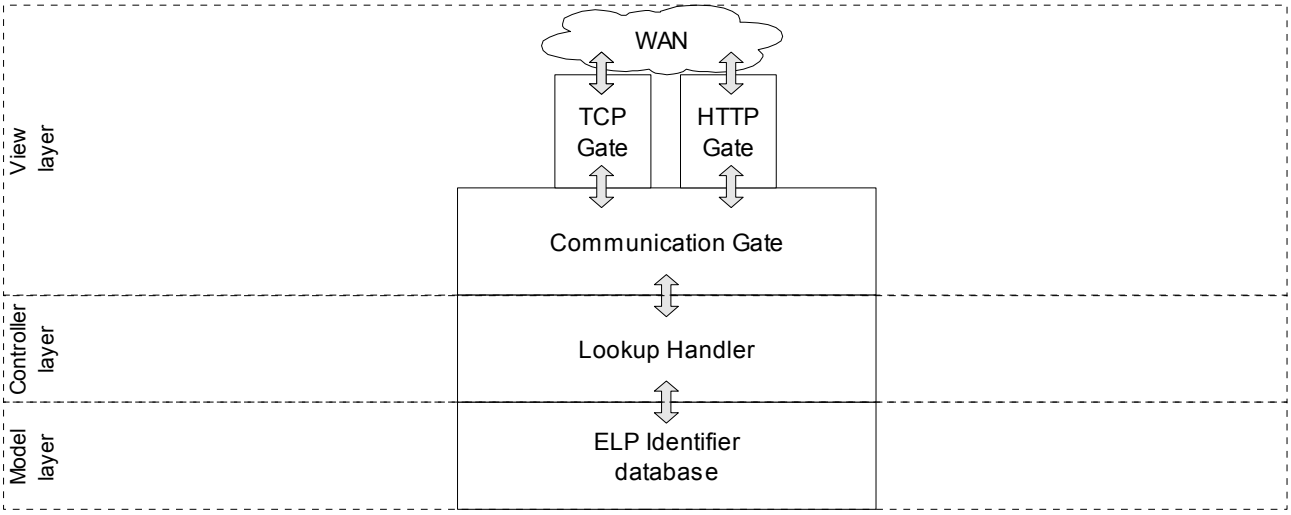


Figure 10.3 – ELP Name Service architecture

An XML formatted lookup message is received by one of the specific transport method gates (TCP, HTTP). This request contains three parameters, namely the ELP identifier, the service identifier and the version of the service. The lookup is then passed through to the communication interface that in its turn passes it through to the lookup handler. The lookup handler checks the syntax and the presence of required fields of the received message. If one of these checks has a negative

result then an error message is sent to the requester. It has to be pointed out that the lookup message always is an XML formatted message containing the three parameters specified by a XML Document Type Definition and that no translation is required.

If a correct lookup message is received then the lookup handler requests the transport methods and their parameters from the database based on the received ELP identifier, service and version. If the database contains no records for the requested lookup then an error message is sent to the requester. If it does contain one or more records, that describe transport methods and their parameters to communicate with the node represented by the ELP identifier, then these records are used to create a reply containing this information. It should always be possible to give a reply to a requester using the gate that the lookup was received from. A TCP or HTTP connection is simply not closed after the lookup has been received and can therefore be used to send the reply. If a lookup request is received using SMTP then there has to be a reply-to e-mail address in the received message. The administration of which lookup request is received from which gate using which transport method parameters is done by the communication gate, but it requires no persistent storage, because a failure of the ELP Name Service during operation will simply result in an error or time-out at the requester. The only possibility for the requester to lookup the ELP identifier is to try again (later).

The ELP Name Service message definitions given in appendix I can be used to give the following example of meta messages sent during an ELP identifier lookup.

The following lookup message is sent to the ELP Name Service including the three required parameters:

```
<lookupRequest>
	<service>ELP</service>
	<serviceVersion>1.0.0</serviceVersion>
	<identifier>gdx</identifier>
</lookupRequest>
```

The following message is the reply of the ELP Name Service. It includes the parameters of the original lookup request for the requester to be able to map the reply to its request. This is required when the lookup is done using some transport method, such as SMTP, that has no connection state that can be used for this purpose.

```
<lookupResult>
	<service>ELP</service>
	<serviceVersion>1.0.0</serviceVersion>
	<identifier>gdx</identifier>
	<resultList>
		<result>
			<transportMethod>HTTP</transportMethod>
			<transportParameters>
				<URL>http://www.somedomain.tld/cgi-bin/elp/elp.cgi</URL>
				<requestMethod>POST</requestMethod>
			</transportParameters>
		</result>
		<result>
			<transportMethod>SMTP</transportMethod>
			<transportParameters>
				<mailTo>elp@somedomain.tld</mailTo>
			<transportParameters>
		</result>
	</resultList>
</lookupResult>
```

From the reply it can easily be concluded that the ELP user identified by ELP identifier 'gdx' supports two transport methods to receive ELP messages.

10.4 ELP Prototype Implementation

Using the MVC pattern, a start as been made to develop a prototype of ELP that supports the key functionality mentioned earlier. The prototype is developed using Borland Delphi that can be considered as an object oriented version of the pascal development language. Global Data Exchange uses a source convention that has many similarities with the MVC design pattern. The source convention is used to split source code into five multi-tier categories:

- Presentation Objects (Po): objects within this category are used by the User Interface
- Data Objects (Do): objects within this category are used for (persistent) storage and communication
- Business logic Objects (Bo): objects within this category are used to process logic without having state
- Entity Objects (Eo): objects within this category represent data entities
- Collector Objects (Co): objects within this category are used to create and link objects of the other categories to form a single functional unit

Two of the five categories, namely Eo and Co, are not present within the MVC design pattern. The main reason for this can be that objects within Eo are used by parts of all layers and that Co is generally the glue to link the three layers. Two of the remaining three categories can directly be mapped to the three layers, namely Po to View and Bo to Controller. The main difference between the categories and the MVC design pattern is the part whereto communication belongs. The MVC design pattern includes this into the View layer, but within the five categories it is part of Do that is comparable to the Model layer.

The prototype consists of seven instances of (inherited) objects (excluding the entity objects):

- **TCoActionManager**: the ELP prototype collector objects. This objects creates, initializes and links most of the other objects.
- **TDoELPStorageFirebird** (inherited from TDoELPStorageBase): the objects that can retrieve/store entity objects from/to persistent storage (Firebird DBMS). Other DBMS can easily be supported by inheriting from TDoELPStorageBase without having to change TCoActionManager.
- **TBoSessionHandler**: the ELP session handler that takes care of sessions used for communication.
- **TBoActionHandlerReplication** (inherited from TBoActionHandlerBase): this action handler takes care of adding and removing nodes to the replication of an object. Other action handlers are all inherited from TBoActionHandlerBase to provide a common presentation to the TCoActionManager.
- **TDoCommExt**: the object that represents the external communication gate. It creates, initializes and links the following Gate and NameService objects.
- **TDoELPGateHTTP** (inherited from TDoELPGateBase): the object that takes care of sending and receiving messages using the HTTP transport method. Multiple transports can easily be implemented by inheriting from TDoELPGateBase.
- **TDoELPNameService**: the object that can lookup ELP identifiers at the central ELP Name Service.

The following figure represents the creation and initialization of each object where an arrow means “creates and initializes”. Despite of communication being part of the Do category, the three layers are clear.

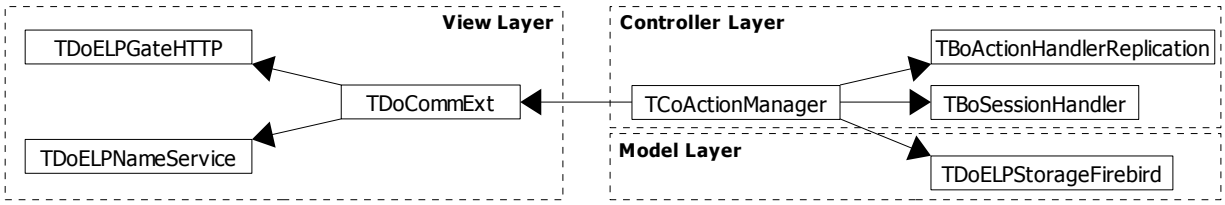


Figure 10.4 – Object Oriented representation of the prototype

The creation and initialization of the objects is done in the order indicated by steps in the following table. If the 'Results in' of step X indicates that it initiates other steps then X is finished after the steps it initiated.

Step	Consist of	Results in
1	Creation and initialization of TCoActionManager instance.	Collector Object that initiates steps 1, 2, 6 and 7.
2	Creation and initialization of TDoELPStorageFirebird instance.	Persistent storage is available.
3	Creation and initialization of TDoCommExt instance.	External communication gate with the message in-queue. It initiates steps 4 and 5.
4	Creation and initialization of TDoELPNameService instance.	TDoCommExt is able to lookup ELP identifiers.
5	Creation and initialization of TDoELPGateHTTP instance.	HTTP server and client. ELP message can be received and enqueued (not yet processed).
6	Creation and initialization of TBoSessionHandler instance.	Action handlers can request sessions that are required for communication.
7	Creation and initialization of TBoActionHandlerReplication instance.	Actions that add/remove nodes from to/from a replication can be performed. It registered the ELP messages it handles at the action manager.
8	The TCoActionManager instance enables its functionality to start processing the in- and out-queue of messages.	ELP prototype is up-and-running.

Table 10.1 – ELP prototype object creation and initialization steps

The eight steps of table 10.1 are performed in such an order that it is not possible for an instance to call methods of an object that was not yet created and initialized. Since this is only a prototype, it is assumed that every component functions and keeps functioning after it is initialized. This assumption can not be made for a non-prototype implementation, for example because a database connection can fail during operation.

Some parts of figure 10.1 are not present in figure 10.4 and are not included because of the simplicity of the prototype. Two parts that are not present are the internal communication gate and the Business process handler. These parts are left out, because no real business process is implemented and the key functions (add/remove a node and update an object) can be simulated by calls from a temporary prototype object that simulates calls that would have been coming from the internal communication gate. Two other part that are not present are the authentication and synchronization handlers. This first is left out, because it is assumed that actions don't require to authenticate themselves at other nodes. The latter is left out, because it is never implemented due to time restrictions although it was originally planned to be implemented. Finally, the TDoELPNameService isn't completely implemented, but supports the lookup of three ELP identifier hard-coded what implies that the ELP Name Service is also not implemented.

As mentioned in the previous paragraph, the prototype is not completely implemented as planned due to time restrictions that would have been exceeded. Nevertheless, the ELP prototype implementation has several working parts. First, all steps of table 10.1 are performed successfully. This results in an application that can send/receive messages (external communication gate) and forward received messages to the correct handler (action manager). The session handler is

completely implemented, including resending of messages, providing a reliable communication layer to the action handlers. The session handler uses the database provided by TDoELPStorageFirebird to store information about, for example, last received messages and session states. The action handler TBoActionHandlerReplication is unfortunately only partly implemented. However, it is implemented so far that it supports sending and receiving communication with two other nodes, using sessions that are provided by the session handler. Since the sending and receiving of at least a few kind of action messages works, it can be concluded that it is possible for all other action handlers to send and receive action messages.

10.5 Prototype retrospect

The first paragraph of this chapter defined five questions of which two are answered before the designing the ELP prototype and three can be answered afterwards. This paragraph is a retrospect on the prototype that focuses on answers to the final three questions.

The primary objective of the prototype was make a conclusion whether the theoretical approaches and conclusions of chapter 7 through 9 can be used to implement the key functionality that consisted of one of the business processes of chapter 6. Unfortunately, the prototype was not finished which resulted in not being able to make this conclusion with a great certainty. However, it is possible to draw some conclusions from the prototype design and the parts that are implemented. The ELP implementation that can be designed and implemented using the knowledge of the ELP prototype chapter is referred to as the full ELP implementation.

It can be concluded that the MVC design pattern is suitable for the implementation of the full ELP implementation. All parts of the prototype can be split into exactly one of the three layers and there are no indications that the full ELP implementation should have a different architecture. Also, the MVC design pattern is suitable for implementations based on the Po, Bo, Do, etc coding conventions that are used by Global Data Exchange.

The illustration of the ELP architecture shows that the parts of the view layer can easily be split vertically into two separate sub-layers, namely the External and Internal Communication Gate. One of the advantages if this is the principle of separation-of-concerns that enables split development, testing and installations. In fact, it possible to split more parts of the architecture what can be used to make a conclusion about the extendability and practical limitations of the architecture. This can be done by splitting every layer into the category 'more than one can exist' or 'only one can exist'. The parts of the architecture that belong to the first category are all parts except those that have state, thus the parts in the model layer. Another part that has practical limitations is the throughput of the “WAN cloud” with trivial reasons and solutions. From this it can be concluded that the ELP architecture is limited by the capacity of the parts in the model layer, although it is clear these parts can be separated physically. To be more precise, the transactions at the storage facilities, that guarantee a single state for each object to the other parts of the ELP architecture, form the bottleneck.

The performance of the ELP prototype could not be measured, because it wasn't finished. The session handler however, is tested with 250 concurrent clients creating and ending sessions. This test showed no performance issues at a AMD Athlon64 3200+ computer with 1.5 gigabyte RAM. From this test and the knowledge about the architecture limitations, it can be concluded that performance limitations do not play a key role as long as the parts of the full ELP implementation are developed as separate units that are, apart from IPC, able to communicate with each other over a network connection.

The implementation result of the prototype cannot be considered as an implementation that already can be used for communication about business processes. The reason for this is that it currently functions as a base that provides communication between nodes. The actual business

processes that should be part of the key functionality are not implemented, because of the absence of the business-, synchronization-, and replication handler. The prototype did show that it is possible to send and receive “action messages” implying that as long as the functionality of all of these handlers is based on synchronous communication there is no indication that the complete prototype implementation would have failed. Unfortunately, the functionality of these handlers is based on the contents of chapters 7 through 9 and these designs and decisions could not be tested using the prototype.

Altogether, no real drawbacks of the ELP prototype design and implementation appeared, although this is doesn't lead to a conclusion about the primary objective of the prototype.

10.6 Summary

The ELP prototype is designed using the Model-View-Controller design pattern separating business logic, persistent storage and user interfaces. The key functionality that one would like to be implemented is to add/remove nodes to/from a replication and to update an object (synchronization). The ELP prototype has to be able to lookup ELP identifiers at the ELP Name Service. The ELP Name Service is also designed using the MVC design pattern and provides the functionality to lookup transport methods for a given ELP identifier. The ELP prototype is made using the source code conventions of Global Data Exchange, separating objects into specific categories, that have many similarities with the MVC design pattern layers.

The implementation of the prototype is not completely finished due to time restrictions. It does however support functionality to send/receive message, build complete sessions and exchange messages used for business processes. The key functionality is not completely implemented, but fortunately time restriction was the only aspect that prevented it from being implemented.

11 Discussion and conclusions

11.1 Introduction

This thesis considered many aspects about the design of an information system that provides functionality to exchange information about the execution of transportation orders. Within this final chapter all findings are drawn together, discussed and a conclusion is made. The first paragraph focuses on the discussion of findings of chapter 2 to 9. Next, an answer is given to the research question of chapter 3 followed by the conclusion of this thesis.

11.2 Discussion of findings

Within chapter 2 it is assumed that outsourcing transportation creates a financial benefit for both client and transport company. This benefit is derived from the more flexible capacity and specialization of a transport company by being able to use transportation means of other companies. The benefits of outsourcing are assumed to be present in all modern industrialized countries around the world. Although this seems to be very reasonable, a remark can be made by the fact that outsourcing involves multiple companies that all would like to make a profit. This implies the total profit made can be higher than if only one transportation company was involved, which casts a doubt on the assumption that a financial benefit is created, especially for the client. However, from the fact that outsourcing is so common within the courier industry it is concluded that the financial benefit for at least these companies is present.

Chapter 3 and 4 focused at the questions of which functional requirements had to be met and whether existing solutions already provided solutions that are suitable. This latter question is answered negatively resulting in design decisions for the new ELP information system. A legitimate question that can be asked is whether the introduction of an additional information is justified, because this increases the number of existing solutions to six what is in contrast with the background of ELP, namely increasing the exchange of information between transport information systems. It seems to be trivial that the more solutions to choose from, the more different “languages” will be spoken, which can in fact decrease the exchange of information. Although this seems to be true, chapter 4 mentioned that there are two major aspects that are not supported by existing solutions, namely accurate information for all participants and support for outsourcing between transportation companies. Exactly these two aspects need to be supported by an information system for the transportation industry to increase the financial and operational benefits. From this it is concluded that a new information system is justified.

Chapter 6 introduced the business processes of transport companies and its application to the use cases of chapter 5. In practice, not every transport company will have identical business processes raising the question whether designing an information system using these processes creates a solution for the majority of transport companies. This is assumed to be true, because the collapsed business processes of paragraph 6.1 are based on those of the existing solutions of chapter 4, such as requesting quotations and placing orders. Some of the existing solutions are mature standards that have proved that these business processes are common for many companies.

One of the parts of this thesis that can seriously be doubted is whether the CDM of chapter 7 is really suitable and as commonly applicable as its name suggests. The question whether the CDM is really suitable as a model that can be used by introducing two-way mapping functions cannot be answered positively or negatively, because this requires more research of existing transport management systems and their data models. At least it is possible to create those mapping functions for both software products of Global Data Exchange enabling the use of ELP and its CDM for the majority of courier companies in The Netherlands. An additional positive aspect of the

CDM is the possibility to extend it with proprietary attributes that introduce more flexibility.

Another interesting part of chapter 7 is the exchange of information and the rules that have to be followed to make sure that every company or information system has up-to-date information. It can be questioned whether these rules are always followed in practice or what happens if they aren't. Although assumed that rules are followed, it can be a part of future work to create some kind of certification or quality mark on implementations of ELP.

Chapter 8 mentioned the assumption of the Internet as communication means. Several reasons are given to indicate why this assumption doesn't hold. However, chances of implementations being based on transport methods that use the Internet seem to be quite high as shown by the prototype of chapter 10 and the transport methods of RosettaNet. This raises the question whether, for example, the routing methods of chapter 8 present useful information for ELP implementations. Although this information is not required to create implementations of ELP that use Internet-based transport methods, the assumption of chapter 9, that a communication network based on the theory of chapter 8 exists, is based on this communication principle.

The comparison of replication techniques in chapter 9 showed that scalability had a best score of mediocre and even lower for some of the techniques. Due to this it can be questioned whether distributed database systems in fact provide a suitable propagation method for ELP. Although the scalability of the techniques is one of the disadvantages, ELP fortunately has a practical limit on the number of companies involved when an order is outsourced. Considering the outsourcing between courier companies in The Netherlands (a few participants), it is not very likely that the number of nodes within a replication exceeds the number where scalability problems can occur.

As far as the abbreviation ELP is concerned, the E of Extendible only plays a small role within the CDM subject. However, if RosettaNet did have some support for custom business processes, the term Extendible was in place for that existing solution. The supported business processes of RosettaNet are very suitable as a basis for ELP and if future research defines the exact ELP business processes, the term Extendible can become more valuable if ELP does supports custom business processes.

11.3 Answer to the research question

This paragraph is dedicated to answering the research question. Although the research question is quite general, it is possible to give an answer using the research that is done and described in this thesis. Before answering the research question, it is repeated:

How can an information system be designed that provides general functionality to exchange information about the execution of the transport of goods and give the possibility to extend it with proprietary elements?

In the research question, three elements can be distinguished. The first element is the functionality to exchange information. The second element is the subject about what information is exchanged and the third element is the possibility to support proprietary elements within this exchange. From this it is clear that 'exchange of information' is the key element in the research question.

The three elements can be used to answer the research question by answering the questions:

- What information is exchanged?
- How is this information exchanged?
- How can this exchange be extended to support proprietary elements?

The information system, as meant by the research question, is called ELP which is an abbreviation for Extendible Logistics Protocol. The information that is exchanged, is information that is required during the business processes of transport companies. To provide an answer to the research question, it has to be clear what the business processes of a transportation company are, especially when orders are outsourced. By examining these business processes, the information that is required to be exchanged can be extracted and modeled into a universal data model that is suitable for many transport companies. This is why chapters 6 is dedicated to the subject of business processes followed by chapter 7 that is dedicated to the Common Data Model.

Having a business processes model and the information that is required to be exchanged, the next step is to actually exchange this information as indicated by the second element. To be able to exchange information there has to be a way to communicate electronically. The main question about communication is which participants communicate with each other when an order is (repeatedly) outsourced? Chapter 8 provides several alternatives that enable participants to communicate, without being limited to one assumption about restrictions on which participants are allowed to communicate with, because solutions are given for many alternatives. The alternatives have different advantages (and disadvantages) on the properties of the complexity of adding nodes, the distance between nodes, the requirement of routing and the publishing of confidential information. There exist correlations between these properties, for example, a low complexity of adding nodes results in a higher distance between them and vice versa. Requirements that cannot be realized at the same time are a low complexity of adding nodes and direct communication channels, a low complexity of adding nodes and a distance of $O(1)$, and only communication with direct business partners and a distance lower than $O(n)$. The introduction of “trusted nodes” and “buffer nodes” increases security and reliability. All alternatives enable the participants to place orders and receive progress information, independent of whether a company would only like to communicate with its direct business partner or not.

It would be too strict to consider the second question only as a way to ask for communication between participants. From the business processes and information to be exchanged, it became clear that in fact many participants are using equal (shared) information that needs to be accurate for all of them to execute their operational business processes as smooth as possible. To provide this accuracy and the ability for each participant to make changes there has to be a technology that provides propagation of updates and the prevention of conflicts. This is why chapter 9 is dedicated to the subject of distributed databases as solution for these requirements. Apart from providing the ability to make changes for every participant, it also provides a solution to prevent participants from making (undesired) changes.

The third question involves an extension to the provided design elements of the information system to exchange information. Unfortunately there exists no chapter that is dedicated to this requirement, except for one paragraph about additional requirements and extension on attributes of entities. However, the messages that are used for the business processes of chapter 6 are not yet defined what enables the possibility to introduce specialized messages for custom business processes. These custom business processes are not limited to, for example, order placing and can therefore also be defined to exchange proprietary elements that are additions to the predefined business processes. Although the third element is not provided by this thesis, the design of ELP anticipated on this by leaving some additional future research on those subjects that could provide it, for example the support for custom business processes and the suitability of the CDM. This implies that this element is not excluded, but only not supported at this moment.

Designing an information system costs a lot of effort and it would be a waste of knowledge if no existing solutions were analyzed. This is why (reusable) knowledge is obtained from a research of existing solutions as provided by chapter 4. This obtained knowledge is used throughout chapters 6 to 10 to use many advantages and avoid many disadvantages. Overall it influenced the design of ELP in such a way that is advised to keep the electronic messages for business processes close to RosettaNets' message for (future) compatibility with one of the preferred existing solutions.

11.4 Conclusion

The transportation of goods appears all over the world by road, rail and air. Especially in countries with a high prosperity many goods are transported from manufacturers to the demanding customers. Within transportation, the manufacturer, one or more transport companies and the receiver are the main participants. Transport companies frequently outsource their orders, because this brings in cost savings, more flexible capacity and the possibility to offer specializations that a company itself can't. These companies use many different Transport Management Software (TMS) products that are available for this large industry. These products enable them to create an increase in efficiency, error prevention and service provided to their customers. The latter is mainly achieved by offering accurate Track & Trace information. Existing software product barely offer functionality to exchange information between transport companies in a common way. The main question that can be asked is: how can the software product be altered to support the exchange of information to also make use of the advantages of TMS products when an order is outsourced?

When outsourcing comes into place there exist business processes at more than one company that require information to be exchanged, which can be done using Electronic Data Interchange (EDI). Research on existing public EDI solutions showed that none of the existing solutions has dedicated support for outsourcing and offering accurate progress information. This implies that none of the existing solutions is suitable to answer the main question. The best suitable existing solution, RosettaNet, doesn't support the possibility to extend it with custom functionality to add these requirements. However, the electronic messages for business processes are suitable and in line with the business processes of this thesis. Due to this, these electronic messages are recommended to be used as initial concept for ELP and be completed where needed. This enables instant support for many common business processes and utilizes the valuable knowledge and semantics of the electronic message of these business processes. In short, it is recommended to re-use as many aspects of RosettaNet as possible for ELP to benefit from the effort and knowledge of the RosettaNet EDI standard.

The electronic messages of RosettaNet don't support all the data that is required to exchange data when EDI is used for outsourcing of transport orders. To take care of this, it is required that these data model requirements are prescribed in such a way they are compatible with as many TMS products as possible. Considering that it is not likely that existing TMS products model the data required for business processes equally, the Common Data Model of this thesis offers a suitable model to ensure that these products represent data in electronic messages in a way that they can all understand. Unfortunately, due to the fact that the CDM design is based on the products of Global Data Exchange and RosettaNet, it is only possible to conclude that it is suitable for courier companies, but not that it is suitable for a wider range of transport companies. This is why additional research needs to be done to see whether its models of orders, goods, track and trace information, etc are suitable to be mapped to and from other TMS product. This research can also show whether the support for extending and requiring attributes is a valuable feature or that it is not suitable in practice and therefore should be ignored.

To provide accurate progress information, fast and reliable communication means are desired. The current penetration of broadband Internet connectivity in western countries is at such a high level that this is considered a decent basis to provide a communication network for ELP. This takes away several technical communication aspects such as routing. Possible (virtual) communication networks of ELP participants can be split into two categories. The first category exists of small communication networks that only consist of order placing/accepting pairs of participants (one-to-one). The second category exists of all participants that can all directly send each other messages (one-to-many, implying that they are aware of each other), because the Internet is considered a suitable communication means. At this point it is not clear whether every participant will accept that communication takes place between all participants due to, for example, confidentiality. This is why, at this point, only the first category is assumed to be acceptable although this has negative

consequences for the progress information provisioning for downstream nodes and possible conflict during updates. Additional research can provide a better view on the acceptance of the second category. This research can also provide information about the awareness of transport companies what technical and operational consequences outsourcing can have, because the business processes exceed company borders.

The supply of accurate progress information to all involved transport companies has many similarities with distributed databases where also all nodes have equal information and changes (progress) are propagated to all other nodes. Since one of the biggest barriers for distributed databases is the increase of the number of nodes (scalability), it is concluded that the first category of communication networks is an advantage for ELP. From the two suitable methods of replication for ELP, lazy replication with master updates method is considered the most suitable. The reasons for this that it is easier to implement and stays closer to the KISS principle. Although two-tier has advantages when mobile nodes exist, it is recommended that they use their limited connectivity to communicate with a buffer node that is part of the ELP lazy-master replication. The knowledge about two-tier can still be used, because two-tier replication can be used between the mobile device and the buffer node.

The ELP prototype introduced confidence about a design that can be used for a full ELP implementation that uses the techniques described in this thesis. This is based on the assumption that all communication is done in a synchronized way. Due to the conclusion that electronic messages should stay close to RosettaNets' electronic messages, this assumption is considered legitimate, because RosettaNet is based on synchronous communication. Another aspect of the ELP prototype that is useful, is the ELP Name Service. The reason for this is that RosettaNet also supports multiple transport methods, in line with the goal of the ELP Name Service.

The designs that are provided in this thesis offer a preliminary solution that can be used to answer the main question. The advantages that transport companies can get from the use of information technology as well as outsourcing can be combined into an even larger advantage. For this to become reality, additional research needs to be done on existing TMS product data models (and their support business processes) as well as on acceptance of (company border crossing) business processes on the subject of communication and operational issues.

12 Future research

Throughout this thesis there exist several subjects that require more attention in the future. Examples of this are parts that are defined to be out of scope for now, possible problems that might occur and additional research that need to be done on uncertainties. This chapter summarizes these aspects that would create valuable additions that can be done in the future.

12.1 Functional requirements and business processes

Table 4.1 of chapter 4 defined a list of functional requirements that have a “low” of “very low” importance and therefore left out of this thesis. This doesn't imply that they can be ignored. A part of these functional requirements are about the negotiation, cancellation and reservation of orders. At this moment these functional requirements are considered not to required by ELP, because it is assumed that an order is always accepted and financial aspects are out of scope. Other functional requirements that are not taken into consideration are those on the subject of management reports and legal issues. To create a mature standard, future work should include these subjects.

Paragraph 6.3.1 introduced the problem of circular quotation requests. A given possible solution that is given, is to use unique identifiers for transportables that need to be transported. This enables transport companies to detect whether a circular quotation request appears. However, at this point in the business process there doesn't exist any 'shared data' of the transportable what can not stop a transport company to use new identifiers for each transportable and treat it as an order that is not yet outsourced. Future research can provide a better technical solution or a set of rules that includes that these unique identifiers are not allowed to be changed.

12.2 The Common Data Model

The CDM is primarily based on the two existing software products of Global Data Exchange. Next, this model is altered to support logistic business processes of companies that are not dedicated to the courier industry. It is assumed that the CDM provides a model that can be used to map information from/to existing proprietary data models. Future research needs to be done to change this assumption to a well founded conclusion.

The business processes of ELP cross company borders and therefore create more uncertainties for companies that outsource orders. These uncertainties are related to operational and technical obligations that participants have. To limit the uncertainties it can be valuable future work to develop a certification or quality mark that can be assigned to implementations of ELP. This can introduce more confidence, because participants' implementations guarantee that correct information is transmitted and the set of rules is obeyed.

12.3 Exchanging information

The Common Data Model provides an extended model to describe information that is exchanged during the business processes. However, this thesis doesn't describe any electronic message aspects that are are used to conduct electronic business. Future research has to define these messages where the messages of RosettaNet can be used as starting point. This research can also include the introduction of messages that can be used to support custom business processes, extending ELP with custom extensions.

Chapter 8 defined two categories for communication. No real conclusion is made about which alternative provides the solution to be used, depending on whether communication with other, not directly involved, participants is acceptable in practice. Future research can provide an answer to this question. Fortunately, all described alternatives provide a communication network that can be used to place orders and receive progress information.

12.4 ELP Prototype

The ELP prototype assumes that it always in a successful state, what means that communication means, resources and databases are always available and function flawless. It is trivial that this assumption doesn't hold when an ELP implementation is used in a production environment. This implies that a full ELP implementation must take this in consideration and support failures during operation. An approach to this can be to introduce a special state property of the Action Manager that is (indirectly) updated and consulted by the other components. For example, this property can be used by components to halt correctly and re-initialize if useful.

13 Appendix A – Subquestions index

The following table contains all the subquestions of chapter 3 together with a reference to a paragraph in which the subquestion is answered and/or related to the context.

Identifier	Subquestion	Paragraph
Q001	Why would users like to exchange information?	4.1.1
Q002	Which business processes are the users involved in?	4.1.1
Q003	What information is going to be exchanged during the business processes?	6.2.1; 7.3.1
Q004	What responsibility during conducting business processes does every participant have and are these responsibilities equally distributed?	4.1.2; 4.1.5; 5.1; 8.3; 8.4.2; 9.1.1
Q005	How valuable is an information system to exchange information to the participants?	2.2
Q006	How is the ownership of information organized?	9.6.2
Q007	What legal aspects, such as confidentiality, authentication and digital signatures, are involved in the business processes?	4.1.6; 8.3; 9.1.1; 9.1.2
Q101	What solutions are currently available?	4.2
Q102	Is there any need for a new information system?	4.2.6
Q103	What is the maturity and acceptance of existing solutions?	4.2.6
Q104	Which properties of existing solutions are desired in a new information system?	4.2.7
Q105	Which desired properties of a new information system existing solutions not provide?	4.2.6; 4.2.7
Q106	How compatible should a new information system be with existing information systems?	4.2.6; 10.2
Q107	What barriers can be expected for a new information system to be accepted?	4.2.1; 4.2.6; 8.3
Q108	Which investments are required for a new information system compared to existing solutions?	4.2.6
Q109	What legal aspects, such as licenses and patents, are involved?	4.2.2
Q201	Is this system only applicable within the transportation industry?	2.2, 3.2
Q202	What extendability can be expected of ELP?	7.2.2
Q203	Is is possible to design the system in such a way that it provides functionality to exchange business process information in general, for example by introducing multiple layers?	4.2.4; 4.2.6
Q204	How can the information system be designed to not strictly limit its participants to standard business processes to increase acceptance and compatibility?	4.2.4; 4.2.7; 7.2.2.2
Q301	What are the requirements for availability, security, accuracy and performance of the exchange of information?	8.2; 8.3; 9.2; 9.3; 9.4.5
Q302	Is it possible that participants do not agree on the information they exchange and how can these conflict be prevented or solved?	9.1.2; 9.3; 9.4.2; 9.4.3
Q303	How can a participant continue to work while not being able communicate with other participants and are there any limitations to this?	9.2
Q304	How can it be prevented that all participants fully rely on the other participants being available?	9.4.3
Q401	Are business processes limited to an exact number of participants?	9.3
Q402	How can participants be added to business processes?	9.6.1
Q403	How is the responsibility organized when a participant would like to add another participant that is unknown to the existing participants?	9.6.2
Q404	How are the rights and relationships between participants managed?	9.6.2
Q405	How is the administration of participants set-up?	9.6
Q406a	Does every participant know about all other participants?	8.4.2
Q406b	Is it required that every participant is able to communicate with all other participants for every business process?	8.4.2
Q501	How is communication between participants set-up?	8.1
Q502	What kinds of communication means are suitable?	8.1
Q503	What are the consequences if the information system fails?	8.2; 9.3; 9.4.5
Q504	Which techniques can be used to exchange information between participants?	9.1.1
Q505	What are the consequences of different locale settings worldwide?	4.2.3
Q506	Is is possible to supply ELP functionality as middleware?	4.2.7; 10.2
Q507	Which existing technological standards can be used to simplify implementations and increase compatibility?	4.2.6
Q508	Are centralized external coordinators needed or can they be avoided?	8.3; 9.6.2

14 Appendix B – EDIFACT and XML message comparison

The following example illustrates the difference in size of an EDIFACT message and a comparable XML message that only consists of the second segment of the EDIFACT message [StylusStudio]:

EDIFACT	XML
UNA:+.? ' UNB+UNOA:3+STYLUSSTUDIO:1+DATADIRECT:1+20051107:1159+6002' UNH+SSDD1+ORDERS:D:03B:UN:EAN008' BGM+220+BKOD99+9' DTM+137:20051107:102' NAD+BY+5412345000176::9' NAD+SU+4012345000094::9' LIN+1+1+0764569104:IB' QTY+1:25' FTX+AFM+1++XPath 2.0 Programmer?'s Reference' LIN+2+1+0764569090:IB' QTY+1:25' FTX+AFM+1++XSLT 2.0 Programmer?'s Reference' LIN+3+1+1861004656:IB' QTY+1:16' FTX+AFM+1++Java Server Programming' LIN+4+1+0596006756:IB' QTY+1:10' FTX+AFM+1++Enterprise Service Bus' UNS+S' CNT+2:4' UNT+22+SSDD1' UNZ+1+6002'	<EDIFACT> <UNB> <UNB01> <UNB0101><!--0001: Syntax identifier-->UNOA<!--UN/ECE level A--></UNB0101> <UNB0102><!--0002: Syntax version number-->4<!--Version 4--></UNB0102> </UNB01> <UNB02> <UNB0201><!--0004: Interchange sender identification-->STYLUSSTUDIO</UNB0201> <UNB0202><!--0007: Identification code qualifier-->1<!--DUNS (Data Universal Numbering System)--></UNB0202> </UNB02> <UNB03> <UNB0301><!--0010: Interchange recipient identification-->DATADIRECT</UNB0301> <UNB0302><!--0007: Identification code qualifier-->1<!--DUNS (Data Universal Numbering System)--></UNB0302> </UNB03> <UNB04> <UNB0401><!--0017: Date-->20051107</UNB0401> <UNB0402><!--0019: Time-->1159</UNB0402> </UNB04> <UNB05><!--0020: INTERCHANGE CONTROL REFERENCE-->6002</UNB05> </UNB> ....

Table 14.1 – EDIFACT versus XML message

The first segment of the EDIFACT message is not present in the XML message, because it describes special syntax characters within the EDIFACT message. The size of the EDIFACT message compared to the size of the full XML message shows that in this example the relation is approximately 9 to 100 (9%).

15 Appendix C – Functional requirements and existing solutions

Nr.	Design decision	Refers to existing solution(s)	Refers to requirement(s)
1.	ELP supports, but should not be limited to, the following standard business processes: 'Request quote', 'Place order' and 'Provide status information'	RosettaNet, EDIFACT, papiNet	RQFuncInf3, RQFuncInf3, RQFuncBus1, RQFuncBus1, RQFuncLeg1
11.	ELP provides a data structure for entities required by the standard business processes	PapiNet, RosettaNet	RQFuncInf1, RQFuncInf2, RQFuncInf3, RQFuncInf3, RQFuncBus1, RQFuncBus1, RQFuncLeg1

Table 15.1 – functional requirements and existing solutions

16 Appendix D - Brief Business Process Modeling Notation

The business process models that are given in this document are based on the Business Process Modeling Notation Specification version 06-02-01 [BPMN]. This part of the appendix only describes the modeling objects that are used within this document and doesn't supply a full description of the BPMN specification. The figures and descriptions are figures and fragments taken from [BPMN]. The copyright of the figures and most of the text fragments belongs to the Object Management Group [OMG].

Events

An event is something that “happens” during the course of a business process. These events affect the flow of the process and usually have a cause (trigger) or an impact (result). Events are circles with open centers to allow internal markers to differentiate different triggers or results. There are three types of Events, based on when they affect the flow: Start, Intermediate, and End.

Object	Description
	**Start** The Start Event indicates where a particular Process will start. In terms of Sequence Flow, the Start Event starts the flow of the Process, and thus, will not have any incoming Sequence Flow—no Sequence Flow can connect to a Start Event.
	**Stop** As the name implies, the End Event indicates where a process will end. In terms of Sequence Flow, the End Event ends the flow of the Process, and thus, will not have any outgoing Sequence Flow—no Sequence Flow can connect from an End Event.
	**Message start event** A message arrives from a participant and triggers the start of the Process.
	**Message intermediate event** A message arrives from a participant and triggers the Event. This causes the Process to continue if it was waiting for the message, or changes the flow for exception handling. In Normal Flow, Message Intermediate Events can be used for sending messages to a participant. If used for exception handling it will change the Normal Flow into an Exception Flow.
	**Timer intermediate event** A specific time-date or a specific cycle (e.g., every Monday at 9am) can be set that will trigger the Event. If used within the main flow it acts as a delay mechanism. If used for exception handling it will change the Normal Flow into an Exception Flow. Within this document is only used to create an Exeception Flow.
	**Compensation intermediate event** This is used for compensation handling--both setting and performing compensation. It call for compensation if the Event is part of a Normal Flow. It reacts to a named compensation call when attached to the boundary of an activity.

Object	Description
	**Cancel intermediate event** This type of Intermediate Event is used within a Transaction Sub-Process. This type of Event MUST be attached to the boundary of a Sub-Process. It SHALL be triggered if a Cancel End Event is reached within the Transaction Sub-Process. It also SHALL be triggered if a Transaction Protocol “Cancel” message has been received while the Transaction is being performed.

Table 16.1 – BPMN events

Activities

An activity is a generic term for work that company performs. An activity can be atomic or non-atomic (compound).

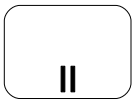
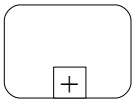
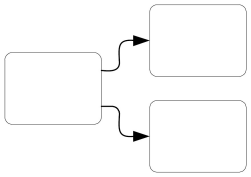
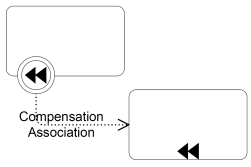
Activity	Description
	**Task Object** A Task is an atomic activity that is included within a Process. A Task is used when the work in the Process is not broken down to a finer level of Process Model detail.
	**Multiple Instances** The task or sub process is executed by multiple instances. The task or sub process must have a descriptor in the upper left corner describing the instances.
	**Collapsed Sub Process** The details of the Sub-Process are not visible in the Diagram. A “plus” sign in the lower-center of the shape indicates that the activity is a Sub-Process and has a lower-level of detail.
	**Forking** BPMN uses the term “fork” to refer to the dividing of a path into two or more parallel paths (also known as an AND-Split). It is a place in the Process where activities can be performed concurrently, rather than sequentially.
	**Associated Compensation** Some activities produce complex effects or specific outputs. If the outcome is determined to be undesirable by some specified criteria (such as an order being canceled), then it will be necessary to “undo” the activities. This is done using a associated compensation that is connected to an Activity using a compensation marker.

Table 16.2 – BPMN activities

Gateways

A Gateway is used to control the divergence and convergence of multiple Sequence Flow. Thus, it will determine branching, forking, merging, and joining of paths.

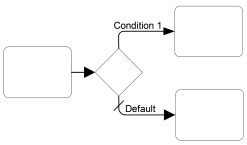
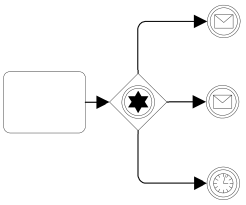
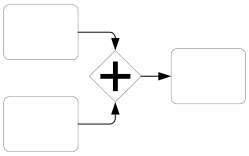
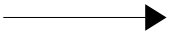
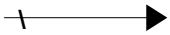
Gateway	Description
	**Exclusive Gateway: Data-Based** This Decision represents a branching point where Alternatives are based on conditional expressions contained within the outgoing Sequence Flow. Only one of the Alternatives will be chosen.
	**Exclusive Gateway: Event-Based** This Decision represents a branching point where Alternatives are based on an Event that occurs at that point in the Process. The specific Event, usually the receipt of a Message, determines which of the paths will be taken. Other types of Events can be used, such as Timer. Only one of the Alternatives will be chosen.
	**Join Gateway** BPMN uses the term “join” to refer to the combining of two or more parallel paths into one path (also known as an AND-Join or synchronization). A Parallel (AND) Gateway is used to show the joining of multiple Flow.

Table 16.3 – BPMN gateways

Sequence Flows

A Sequence Flow is used to show the order that activities will be performed in a Process.

Sequence Flow	Description
	**Normal Flow** Normal Sequence Flow refers to the flow that originates from a Start Event and continues through activities via alternative and parallel paths until it ends at an End Event.
	**Conditional Flow** A Sequence Flow can have condition expressions that are evaluated at runtime to determine whether or not the flow will be used. If the conditional flow is outgoing from an activity, then the Sequence Flow will have a mini-diamond at the beginning of the line (see figure to the right). If the conditional flow is outgoing from a Gateway, then the line will not have a mini-diamond.
	**Default Flow** For Data-Based Exclusive Decisions or Inclusive Decisions, one type of flow is the Default condition flow. This flow will be used only if all the

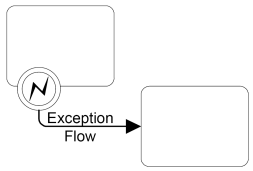
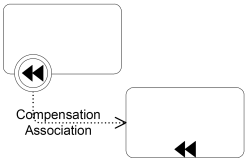
Sequence Flow	Description
	other outgoing conditional flow is not true at runtime. These Sequence Flow will have a diagonal slash will be added to the beginning of the line.
	**Exception Flow** Exception Flow occurs outside the Normal Flow of the Process and is based upon an Intermediate Event that occurs during the performance of the Process.
	Message Flow A Message Flow is used to show the flow of messages between two entities that are prepared to send and receive them. In BPMN, two separate Pools in the Diagram will represent the two entities.
	**Compensation Flow** Compensation Association occurs outside the Normal Flow of the Process and is based upon an event (a Cancel Intermediate Event) that is triggered through the failure of a Transaction or a Compensate Event. The target of the Association must be marked as a Compensation Activity.

Table 16.4 – BPMN sequence flows

Other Objects

The following objects do not fit into one of the previous categories.

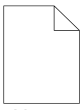
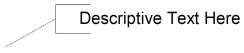
Object	Description
	**Data Object** Data Objects do not have any direct effect on the Sequence Flow or Message Flow of the Process, but they do provide information about what activities require to be performed and/or what they produce.
	**Text Annotation** Text Annotations are a mechanism for a modeler to provide additional information for the reader of a BPMN Diagram.
	**Pool** A Pool represents a Participant in a Process. It is also acts as a “swimlane” and a graphical container for partitioning a set of activities from other Pools, usually in the context of B2B situations.

Table 16.5 – BPMN other objects

17 Appendix E – CDM illustrations

The CDM illustrations in chapter 7 use several elements that represent entities, their relations and attributes. The basic elements that are used are described in this appendix to promote correct interpretation of the ER diagrams.

An entity is drawn using a yellow rectangle that contains the name of the entity. There are two kinds of entities, namely strong and weak entities. Weak entities are displayed using light yellow rectangles:

Figure 17.1 – CDM entities

The relationship between two entities is given using a purple diamond shape rhombus. The rhombus contains a verb that describes the relationship between the two relations. To be sure that the verb is understood correctly, the relation between two entities is read from left-to-right and from top-to-bottom.

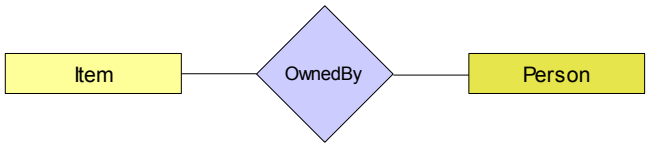


Figure 17.2 – relations between CDM entities

In the examples above is a relation displayed that should be read as 'An item is owned by a person.' where an item cannot exists without the ownership by a person. The number of relations, such as one-to-many, is given using numbers next to the entities outgoing lines. The following figure illustrates that an item is always owned by exactly one person and that one person can own zero or more items. The figure also illustrates how attributes are drawn using orange ovals and primary keys using a bold and underlined font. When an attribute is a discriminating attribute of a weak entity set then it is displayed bold and underlined with a dotted line.

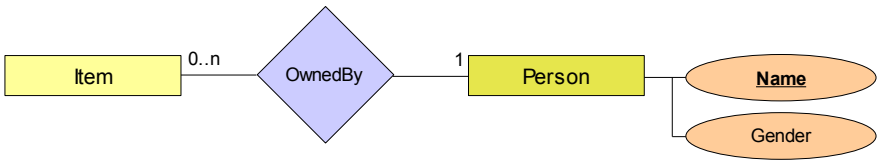


Figure 17.3 – attributes of CDM entities

All entities, attributes and relationships can be found in appendix J where they are described in more detail like type information, limitations and default values. Each entity and attribute is a data structure, whereas a rhombus figure tells how those data structures refer to each other. For example, a one-to-many relation will result in a list data structure to enable multiple references to instances of an equal data structure.

18 Appendix F – Rules for outsourcing and exchanging data

Chapter 7 described outsourcing using transportables, holders and transholders. These data elements and (multilevel) outsourcing can be combined into a list of rules. These are as follows:

(1) A transportable is always contained in one holder. The initial holder is the origin. The final holder is the destination.

(2) A holder can be contained in another holder.

(3) A holder can contain zero or more transportables and/or holders.

(4) A holder can be a transportable.

(5) If a holder is also a transportable (transholder) then the holder hasn't got a fixed location.

(6) The location of a transportable is always the location of the holder by which it is contained in.

(7) If a holder is contained in another holder then the location of the holder is always the location of the holder by which it is contained.

(8) If a transholder is delivered at the final location then the holder of the transportables that are contained in the transholder is changed to the holder that contains the transholder.

(9) If (a part of) the transport of a transportable is outsourced then it should be assumed that the transport is done successfully.

(10) If the Remaining Track (RT) is outsourced to a single transport company then this transport company is responsible for the pick-up at the current location and the delivery at the destination.

(11) If the 'From' location of a TransportableTrack of a transportable is a location of a holder that is owned by a transport company then that transport company can assume that, if the holder of the transportable is not already that holder, the transportable will be delivered at that location and therefore be contained in that holder.

(12) If the Remaining Track is outsourced to multiple transport companies then the linked tracks of these transport companies form the RT. Every transport company is responsible for their part of the RT.

(13) The transport or a part of the transport of a transportable can be outsourced to another transport company whereby all the data of the transportable is supplied to the executing company. Both transport companies can identify the transportable with the same identifier.

(14) If a transport company accepts the (outsourced) transport of a transportable from another transport company then this transport company could also outsource the transport.

(15) If (a part of) the transport of a transportable is outsourced then there is a synchronization mechanism that makes sure that the shared data of the transportable is synchronized if one of the companies changes data of the transportable. Every company has to be part of this synchronization mechanism and is obliged to use it.

(16) If a transport company changes the holder of a transportable to a transholder then this transport company has the knowledge that the transportable is contained in the transholder. If the location of the transholder changes then this company also changes the data of the transportable.

These rules are used to create a description of outsourcing that refers to them to illustrate why every rule is needed. This explanation is graphically illustrated by figure 7.9 because the explanation itself can be confusing.

☞ “transport company” is abbreviated to “company”.

A transportable of a customer, that will be identified as transportable-1, needs to be transported from the customer to a destination. Initially the holder of transportable-1 is the origin (1). The customer places an order at company A to transport the transportable. A truck drives to the customer to pick-up the transportable. At the moment that transportable-1 is picked-up, the holder of transportable-1 changes to the truck and, because of (6), the location of the transportable is the same as the location of the truck.

The next step is the arrival of the truck at the warehouse of company A. The transportable is now unloaded from the truck and stored in the warehouse which affects a change of the holder of the transportable. The holder of the transportable changes from the truck to the warehouse.

A manager at company A decides to outsource the remaining transport to the destination (13). Company A has several transportables in its warehouse that need to be delivered in the same area as transportable-1. The transportation planner at company A decides to send all these transportables on a pallet to a local delivery company in that area: Company C. Company C should do the transport from their warehouse to the destination. Company A places an order at company C to transport transportable-1 from their warehouse to the destination. Due to (9) and (11) company C waits for the transportable to arrive in their warehouse.

Because of (15), both Company A and Company C share data about transportable-1. If one of these companies changes data of transportable-1 then it is updated at the other company (15). This implies that if company C delivers the transportable then company A knows about it. If the customer would like to know some information about the delivery process then he could just get in contact with company A to get this information. The customer can't see that the transport is outsourced and company A always has recent information about the transport of transportable-1.

Company A knows that the next step is to make sure that transportable-1 is delivered at the warehouse of company C to accomplish (12). The pallet, that contains transportable-1, is a holder that is going to be sent to company C (4) (5). This makes the pallet a transportable and a holder, i.e. a transholder. At the moment that transportable-1 is put upon the pallet, the holder of transportable-1 changes from the warehouse to the pallet (1). The holder of the pallet is the warehouse (2) and therefore the location of the transportable is the location of the warehouse (6) (7). Transportable-1 is currently on the pallet in the warehouse of company A. The pallet is identified by Transportable-2. Company C knows (15) that transportable-1 is contained in transportable-2.

Company A decides not to transport the pallet to company C their selves but to outsource the transport to company B. Company B will transport transportable-2, that contains transportable-1, from company A to company C which accomplishes (12). Company A places an order at company B to transport transportable-2 to company C. Now both company A and company B share data about transportable-2 (15).

Company B picks-up transportable-2 at company A. The holder of transportable-2 changes from the warehouse of company A to the truck of company B. Because of (16), the location of transportable-1 is changed because the location of transportable-2 changed. This is done by company A because this company knows that transportable-1 is contained in transportable-2. The change of the location could be mentioned by company C because of (15). It can be concluded that every change of the location of transportable-2 will lead to a change of the location of

transportable-1 and can be mentioned by all companies that share the data of transportable-1.

Transportable-2 is stored at the warehouse of company B and later transported to company C. These steps also involve changes of the holder and the location of transportable-2, but this is not described in detail as this is analogue to the previous paragraphs.

Finally transportable-2 arrives at company C, which is the final destination of transportable-2. Company C knows the holder of transportable-1, namely the just arrived transportable-2. Due to (8) the holder of transportable-1 changes to the warehouse of company C and this accomplishes (11). Company C changes the holder of transportable-1 to the warehouse of company C and (15) implies that this change will be known by company A. If transportable-1 is delivered at the destination then the holder will change to the destination and a POD will be supplied. This information is also known at company A (15) so company A knows that the transport is done.

19 Appendix G - Routing methods in detail

Paragraph 8.4 referred to several routing methods that this appendix describes in more detail.

Flooding

The primary aspects of the flooding method is that every incoming message is sent to every other communication channel except the one the message is received from. If cycles exist in the network then it is possible that a message is sent around forever. To prevent this from happening a flood damping method is needed, such as a hop counter contained in every message that is increased by one when it passes a node. The message is not routed anymore if a maximum hop count is reached. Flooding guarantees that the shortest path is used, because every possible path is used. This also implies that the chance of successful delivery is very high, because, if enough paths exist, the message will even arrive if suddenly one path fails. However, the huge amount of message sent across the network is a price that has to be paid and it also doesn't make it suitable for every situation.

Distance-vector routing

This method introduces a weight function, such as delay, for every (indirect) communication channel from one node to another. First, every node determines the weight, that is put in its routing table, to every neighbor. This information is sent to every neighbor at certain intervals. The receiving node inserts this information into its routing table by updating existing routes and adding new ones. This updated routing table is also sent to its neighbors et cetera. At a certain time, this results in every node knowing one or more paths to every other node including the route to choose with the lowest weight.

The number of messages is certainly lower than that of the flooding method and it also results using the best path, i.e. the path with the lowest weight. However, it also has a disadvantage, namely that the news of a failing path is not spread very quickly. This can lead to the 'Count to Infinity' problem [Doyle] where, in case of a failure, a node uses an alternative route. This node doesn't know that it is still on the best route paths of one of the nodes providing an alternative route. However, the alternative route only has a bit higher weight and so it changes its route table. This results in the nodes on the alternative route to also increases their weight to the failed node. This process continues 'forever' and thus explaining the name of the problem. A limit on the weight can stop this process.

Link state routing

This method uses properties of both flooding and distance-vector routing. First, every node constructs a table containing only its neighbors and their weight function result. This table is spread to all other node in the network by using flooding. Every node is now quickly able to calculate an optimal route to every other node in the network and this information is updated frequently. The quick distribution of routing information and the knowledge about the whole path prevents the 'Count to Infinity' from happening. The number of packets sent using this method is less than that of flooding, because only routing information is flooded instead of every message. Apart from the higher number of messages required, there also exists another disadvantage, namely that of aging of the routing information being flooded. All routers ignore duplicates or older versions of routing information they received from a specific node, where the version is represented by a counter. When a router is rebooted then it starts again at version 0. If no automatic aging method is used then the other routers will accept new routing information only after the same period of time the router was running before it was rebooted; this usually is a long time. An automatic aging method speeds up this process by putting a TTL (time to live) on every piece of routing information received. Routing information from a rebooted router is ignored until the TTL has been reached of its old routing information. This implies that normally every router has to flood routing information again within the TTL to prevent its routing information from being deleted by other routers.

Hierarchical routing

Hierarchical routing is a method that can be put on top of the two previous routing methods. This method divides a large network of communicating nodes into more small ones that are connected. This connection exists between an 'exit' and an 'entry' node (or vice versa depending on the message destination) of two different networks. The advantage of this method is that every node has a smaller routing table, because within one smaller network, each node only has to know the 'exit' node that has the best route to another smaller network. Disadvantages of this method are the possible longer paths and the higher traffic load between an 'exit'-'entry' pair.

Instead of only creating a hierarchy on one level, it is also possible to introduce multiple levels where every level consists of several smaller networks. This can easily be illustrated by the Dutch telephony number assignments. Utrecht is a big city where all telephone numbers start with 030. Several villages around Utrecht also have a telephone number that start with 03, but followed by a 1, 2, 3, etc and are connected to Utrecht. The numbering plan around Eindhoven is analogue using telephone numbers that start with 04. Here its possible to identify multiple small networks that appear on the same level but are not connected, namely the 03(...) and the 04(...) networks. One network level higher, these networks are connected to each other. If a citizen from a village around Eindhoven makes a phone call (sends a message) to a number starting with 033, the phone call is routed through, sequentially, Eindhoven, Utrecht and the village near Utrecht. In a network with N routers, the optimal number of levels is $ln(N)$ where the average increase of the path length can be neglected [Tanenbaum].

Multidestination routing

This method of routing distinguishes itself by messages that can have more than one destination. A node that would like to send a message with the same contents to more than one destination simply adds these destinations to it. If a router receives this message then it sends a copy of it to all the outgoing communication channels for which it knows that at least one of the destinations can be reached. If multiple destinations can be reached by the same outgoing communication channel then these destinations are added to the message before it is sent. For example, if a router has two outgoing communication channels to respectively nodes D,E,F and K,L,M and it receives a message with destinations D,E,F,K,L and M then it makes two copies with destinations D,E,F and K,L,M respectively. Multidestination routing is in fact an addition to another routing method, such as distance-vector routing, to support messages with multiple destinations.

The multidestination routing method assumes that there already exists a network that can be used for sending messages to a single destination, e.g. a network with distance vector routing. The advantage of introducing the multidestination routing is that it reduces the number of messages across the network when a node frequently has to send message to more than one destination. It is trivial that sending these message to the destinations separately involves more messages.

20 Appendix H - 2PC node extension (two-tier replication)

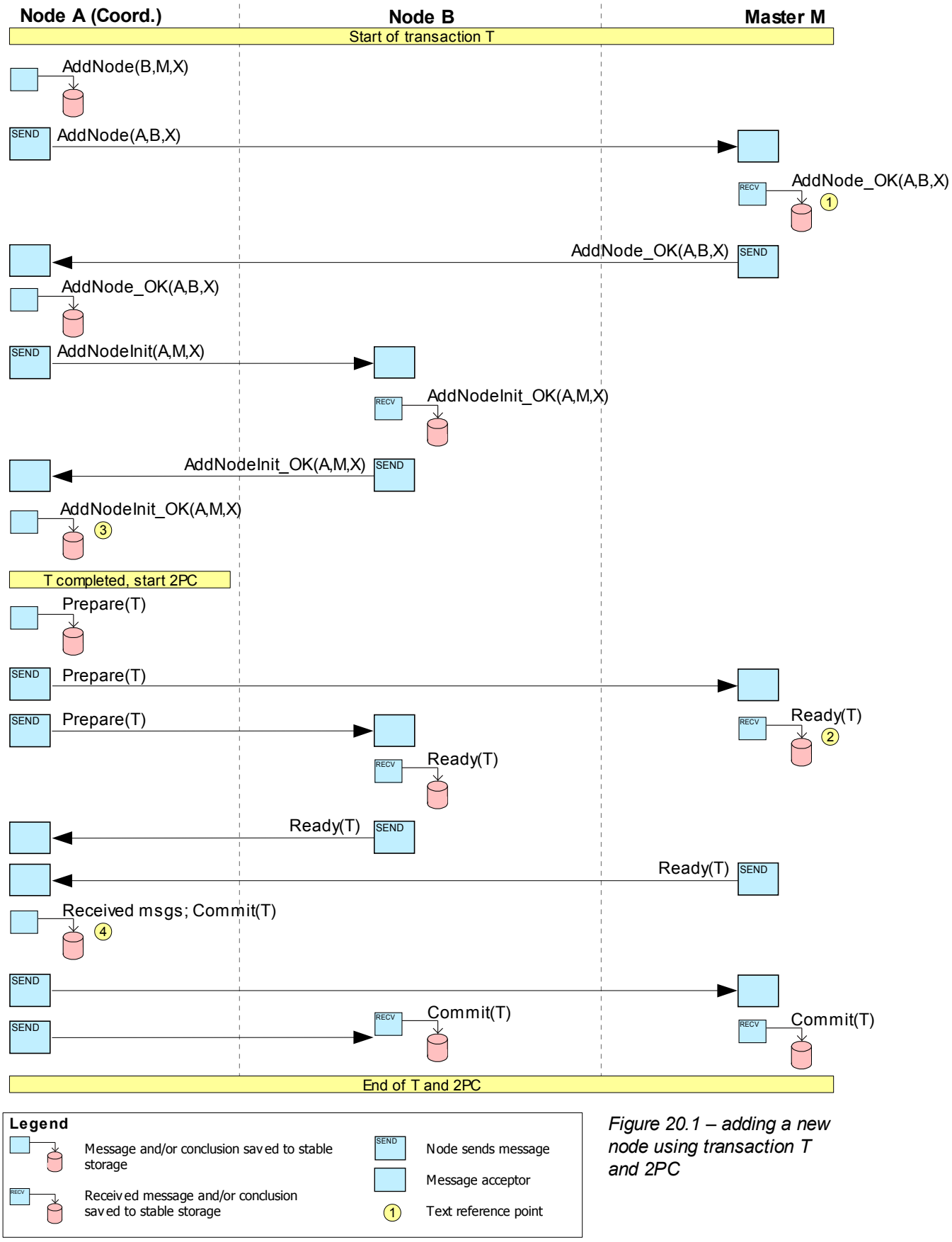


Figure 20.1 – adding a new node using transaction T and 2PC

Figure 22.1 contains the actions performed by the involved nodes where the order is from top to bottom. Transaction T consists of the following messages:

AddNode(A,B,X): Request from A to the master to add B to the replication of X
AddNodeOK(B,M,X): Agreement on the AddNode request where X is the accurate value of X
AddNodeInit(A,M,X): Request to the new node from A to add X mastered by M
AddNodeInit_OK(A,M,X): Agreement on the AddNodeInit request adding the value of X
Prepare(T), Ready(T), Commit(T): Messages used by 2PC

First, node A checks whether M agrees on adding B to the replication of X. Master M replies successfully to this using AddNodeOK that includes the most accurate value of X. This information is saved to stable storage by A. The master stored the version of M to stable storage to know that this is the version sent to node A. Next, node A checks whether B agrees on becoming part of the replication of X mastered by M. Node B responses successful to this what is also saved to stable storage by A. At this point, all nodes agreed on transaction T (the addition of B) and M knows the version of B. These are exactly the properties of the conclusion.

The figure illustrates a successful completion of transaction T, but T can also fail. From reference point 1 object X is locked by M to ensure that A receives the accurate version and that X cannot be removed meanwhile by another transaction (isolation property of the ACID properties). The lock on X cannot exist too long, because it might be blocking other transactions. Master M is able to abort (and rollback) T immediately until reference point 2 in which it stores its decision to follow the final decision of T made by A. Locks at the other nodes are ignored, because updates can only be performed by the master.

At reference point 3, node A knows whether B and M agree on T. If either of those responded earlier with a 'disagree' message then node A could abort (and rollback) T at all location by sending an abort message at any time during T. It is also possible that A aborts the at reference point 3 even if both nodes agreed on T. After reference point 3, the decision of A is stored to stable storage and starts the commit of T by all nodes using 2PC.

The 2PC protocol starts with requesting whether all sites are (still) willing to commit T. The replies to these requests are used to make the decision whether T is going to be committed. If one of the nodes responded with an Abort(T) message to the Prepare(T) message then A sends an Abort(T) to all nodes aborting T. If all of the nodes responded with a Commit(T) message then A can decide to commit or abort T until reference point 4 in which it decision is saves to stable storage. This implies that the fate of the transaction has been sealed. This is the reason why node B and master M store their decision to follow A (at reference point 2). The Commit(T) messages after reference point 4 could also have been Abort(T) messages if A decided to abort T.

The 2PC has a blocking problem when then transaction coordinator, node A, fails at a certain point [Silberschatz]. This point is when it fails after saving Ready(T) to stable stores by B and M and before receiving Commit(T) or Abort(T) by one of those. These nodes are now waiting for the final decision of A, that will never be made or received, while they are not allowed to abort T locally at this point. One solution to this problem is using Three Phase Commit (3PC) in which a new transaction coordinator can be assigned dynamically if B and M are still available and able to communicate with each other. An analogue solution might be to put an expiration-period on T after which M becomes the new transaction coordinator if T reached a blocked state. If A eventually becomes available again after the expiration-period has passed then it has to request the fate of the T at M.

A final remark has to be made for the situation in which only two nodes are involved, for example the master and a node it would like to add. This situation will especially occur within the first group that follows the order scheme. The described method using T and 2PC will now consist of two nodes an becomes less complex, but still suitable for the addition.

21 Appendix I - ELP Name Service message definitions

The following example is based on the message definitions on the next page.

The following lookup message is sent to the ELP Name Service including the three required parameters:

```
<lookupRequest>
	<service>ELP</service>
	<serviceVersion>1.0.0</serviceVersion>
	<identifier>gdx</identifier>
</lookupRequest>
```

The following message is the reply of the ELP Name Service. It includes the parameters of the original lookup request for the requester to be able to map the reply to its request. This is required when the lookup is done using some transport method, such as SMTP, that has no connection state that can be used for this purpose.

```
<lookupResult>
	<service>ELP</service>
	<serviceVersion>1.0.0</serviceVersion>
	<identifier>gdx</identifier>
	<resultList>
		<result>
			<transportMethod>HTTP</transportMethod>
			<transportParameters>
				<URL>http://www.somedomain.tld/cgi-bin/elp/elp.cgi</URL>
				<requestMethod>POST</requestMethod>
			</transportParameters>
		</result>
		<result>
			<transportMethod>SMTP</transportMethod>
			<transportParameters>
				<mailTo>elp@somedomain.tld</mailTo>
			<transportParameters>
		</result>
	</resultList>
</lookupResult>
```

From the reply it can easily be concluded that the ELP user identified by ELP identifier 'gdx' supports two transport methods to receive ELP messages.

ELP Name Service message definitions and technical details

There exist three different message that can be identified by their different root elements. These root elements are lookupRequest, lookupResult and lookupError.

Messages

- Message: lookupRequest

Element	Type	Default	Multiple	Remarks
service	Simple ASCII string, maximum length is 128 characters	“”	No, single occurrence required	The identifier of the service. Currently statically defined as “ELP”.
serviceVersion	Three decimals (>=0, <=255) seperated with a single dot.	1.0.0	No, single occurrence required	The version of the service. Currently statically defined as “1.0.0”.
identifier	Simple ASCII string, maximum length is 128 characters	“”	No, single occurrence required	The ELP identifier

- Message: lookupResult

Element	Type	Default	Multiple	Remarks
service	Simple ASCII string, maximum length is 128 characters	“”	No, single occurrence required	The identifier of the service. Currently statically defined as “ELP”.
serviceVersion	Three decimals (>=0, <=255) seperated with a single dot.	1.0.0	No, single occurrence required	The version of the service. Currently statically defined as “1.0.0”.
identifier	Simple ASCII string, maximum length is 128 characters	“”	No, single occurrence required	The ELP identifier
resultList	Element node 'resultList'	N/A	No, single occurrence required	Result list container element

- Message: lookupError

Element	Type	Default	Multiple	Remarks
message	Simple ASCII string, maximum length is 255characters	“”	No, single occurrence required	Describes the error occurred.
number	One decimals (>=1, <=255) identifying the error	N/A	No, single occurrence required	Error number (1=Syntax Error, 2=ELP Identifier not found)

Elements

- Element: resultList

Element	Type	Default	Multiple	Remarks
result	Element node 'result'	N/A	Yes, at least one required	Single result container element

- Element: result

Element	Type	Default	Multiple	Remarks
transportMethod	Simple ASCII string, maximum length is 64 characters	“”	No, single occurrence required	Transport method identifier
transportParameters	Element node 'transportparameters'	N/A	No, zero or one occurrence possible	Specific element describing required information for the transport method

- Element: transportParameters

The transportParameters element consists of elements that are specific for the value of the transportMethod element. This elements in fact consists of another XML based document. Two possibilities are given below.

Elements of transportParameters when the value of transportMethod is ”HTTP”

Element	Type	Default	Multiple	Remarks
URL	Simple constant String, maximum length is 2048 characters	“”	No, single occurrence required	Only “http” and “https” schemes are valid.
method	Simple constant String, based on RFC 2616 request methods	POST	No, single occurrence required	“POST” and “GET” are the only valid methods. “POST” is preferred.

Elements of transportParameters when the value of transportMethod is ”SMTP”

Element	Type	Default	Multiple	Remarks
mailTo	Simple constant String, based on RFC 2822 addr-spec specification	“”	No, single occurrence required	E-mail address

TCP/IP gate

The ELP Name Service service provider can be reached using TCP/IP on the Internet. The default port of this service is 6150. The host that provides the service is currently not defined as a temporary implementation is used.

22 Appendix J - CDM Entity Details

This appendix provides detailed information about the entities and attributes that are mentioned in chapter 7.

22.1 Base types and requirements

Every data structure of an entity is defined by a table that describes its attributes together with their types, information about being required, default value and a description. First the base types and possibilities for the 'required' columns used are defined.

Type	Description
PosInt	Positive integer within the range of 0 to 2^{31}-1
Int	Integer within the range of -2^{31} to 2^{31}-1
NegInt	Negative integer within the range of -2^{31} to -1
Float	32bit floating point value
String[x]	String of maximum length x characters (Unicode)
Boolean	Derived from Int. 0 is False, True otherwise.
UUID	Universally Unique Identifier (OSF, version 4)
ISO-3166-1	Two character ISO country code (alpha-2)
ISO-8601	Date and/or time based on ISO-8601 standard. If a type refers to ISO-8601 then it must indicate if date and/or time is/are mentioned. A date is represented as: **YYYY-MM-DD** Example: 2006-04-16 A time is represented as: **hh:mm:ss** Example: 13:04:10 The combination of date and time is represented as: **YYYY-MM-DDThh:mm:ss** Example: 2004-04-16T13:04:10 All dates and times are in UTC and therefore no time zones are needed and daylight saving time is not taken in consideration what means that all times are in standard time.
List of ...	An ordered list of the type indicated. The order is indicated by the 'Index' attribute of type Int starting at 0.

The values of the 'required' columns of entities have one of the following abbreviations:

M: Mandatory
The attribute is mandatory and needs to be present. If an attribute is mandatory then the default value is absent.

O: Optional
The attribute is optional and the user is free to provide this data of this attribute.

A: Advised
The attribute is optional, but is it is highly recommended to provide the data for this attribute, even when this implies the use of extra resources.

C: Conditional
The presence of this attribute depends on the presence or value of another attribute of the data structure. The condition is given in the description column and must be able to be processed by an automated system.

22.2 Client entity

Term
A natural person or a company.

Definition
An entity that describes a person or institution created to conduct business or non-profit activities.

Description
A client would like to have transportables transported from one or more origins to one or more destinations. A client can be a natural person, a company or an institute. A client requests a service or product by placing an order at a company.

If the client is a company then it can also be a transport company that is outsourcing its business. This is also the case if the company is a transportation broker. The transportation broker becomes the client of a transport company or even another transportation broker.

Data structure

Attribute	Type	Req	Default	Description
ID	UUID	M	N/A	Identifier of the client.
Name	String[50]	M	N/A	Name of the client.
Address	Address	M	N/A	Address of the client.
Connectivity	Connectivity	M	N/A	Connectivity of the client.
CommerceNumber	String[24]	C	“”	Commerce number issued by the Chamber of Commerce. If CommerceCity or CommerceCountry is given then CommerceNumber has to be given.
CommerceCity	String[100]	C	“”	City where the Chamber of Commerce is settled. If a CommerceCity is given then CommerceCountry has to be given as well.
CommerceCountry	ISO-3166-1	C	“”	A country code of two characters. If a CommerceCountry is given then CommerceCity has to be given as well.
TaxCode	String[24]	O	“”	Tax code issued by the government
ContactPerson	String[50]	O	“”	The contactperson at the company

22.2.1 Additional attribute types of Client

Address

Term
Address

Definition
An entity that describes a fixed location on earth.

Description
An address expresses is a fixed location of a home, business or other building on the earth.

Data structure

Attribute	Type	Req	Default	Description
Street	String[200]	M	N/A	A streets' name
Number	String[20]	M	N/A	A number at the street
Premise	String[200]	O	“”	A premise
PostalCode	String[50]	O	“”	A postal code
City	String[100]	M	N/A	A city
State	String[100]	O	“”	A state
Country	ISO-3166-1	M	N/A	A country

Connectivity

Term
Connectivity

Definition
Connectivity contains information how to reach a business or natural person using telecommunication.

Description
Connectivity contains information that can be used to communicate with another party over a distance.

Data structure

Attribute	Type	Req	Default	Description
Email	String[320]	O	“”	An email address with full domain qualifier
Phone	String[16]	O	“”	A phone number as recommended by ITU-T E.164 including the + prefix
Fax	String[16]	O	“”	A phone number as recommended by ITU-T E.164 including the + prefix
Cellphone	String[16]	O	“”	A phone number as recommended by ITU-T E.164 including the + prefix
Website	String[2048]	O	“”	An URL including the scheme name (e.g. 'http')

22.3 Order entity

Term
Order

Definition
A request from a client to a transportation company or transportation broker to transport transportables from one or more origins to one or more destinations.

Description
An order is sent from a client to a transportation company or transportation broker. The client would like to have some transportables transported from one location to another by the transportation company. The transportables are picked up at one or more origins and are transported to one or more destinations. The client has to present information about the origin(s) and destination(s). Also, the client has to give properties of the transportables that have to be transported. These properties can include size, weight, time aspects, etc. See the definition of transportables for all properties of a transportable.

Data structure

Attribute	Type	Req	Default	Description
ID	UUID	M	N/A	Identifier of the order
ClientID	UUID	M	N/A	Identifier of the client of the order
Moment	ISO-8601 (date and time)	M	N/A	The moment in time that the order is received.
ReferencePerson	String[100]	O	“”	The name of the person that acts as a reference at the client. If the transportation company has any questions about the order then this person acts as contactperson. The value of this attribute is usually the attribute 'ContactPerson' of the Company data structure.

22.4 Transportable entity

Term
Transportable

Definition
Transportables are all physical materials that can be transported from one location to another location.

Description
One transportable can contain one or more physical materials also referred to as goods. They can be in every form, like raw, fluid, packet, parcels, etc. All transportables finally have a POP and POD which can also be 'unknown'.

A transportable always has one origin and one final destination. A transportation company doesn't per se pick up the transportable at the origin and deliver it at the final destination, i.e. the pick up and the delivery is not done by the same transportation company. A transportable can be transported through a track of locations. Every transportable therefore has a tracking list of locations where it has been stored (see holder). Each of these intermediate storage activities can involve a POD.

Data structure

Attribute	Type	Req	Default	Description
ID	UUID	M	N/A	Unique identifier
Holder	UUID	M	N/A	The holder where the transportable is stored in or transported by
PartsQuantity	Integer	O	1	Describes the number of parts in the transportable
Packing	TransportablePacking	M	N/A	Type of packing
LocationOrigin	RouteLocation	C	NULL	The location that gives the origin of the transportable. At least the LocationOrigin or the LocationReturn must be given. If both attributes are given then the LocationReturn prevails for return.
LocationReturn	RouteLocation	C	NULL	In case the transportable could not be delivered, it has to be returned to this location. At least the LocationOrigin or the LocationReturn must be given. If both attributes are given then the LocationReturn prevails for return.
LocationNext	RouteLocation	O	NULL	The next location where the transportable will to be delivered. This could be a intermediate location. If this attribute is not given then it is not known what the next location will be, i.e. it can be the final destination but that it not sure.
LocationFinal	RouteLocation	M	N/A	The location where the transportable finally needs to be delivered.
Dimension	TransportableDimension	O	NULL	A data structure that tells the dimensions of the transportable.
ADR	ADRInformation	O	NULL	A data structure that gives information about dangerous goods.
TemperatureRange	TemperatureRange	O	NULL	A data structure that gives a temperature range wherein the transportable needs to be transported
Priority	Int	M	0	Priority in a range of -100 to +100. -100 is the lowest possible priority, 0 is normal priority, +100 the highest priority
Comment	String[1024]	O	“”	Arbitrary comment for the transportable
ContentDescription	String[1024]	O	“”	Description of the contents of the transportable
ContentValue	Float	C	“”	Value of the content of the transportable. If this attribute is present then the attribute ContentValueConcurrency needs to be present also.

Attribute	Type	Req	Default	Description
ContentValueCurrency	ISO-4217 Code	C	“”	Three characters code of the currency of the content value. If this attribute is present then the attribute ContentValue needs to be present also.
Reference	String[255]	O	“”	A reference for the transportable that is usually defined by the sender or the receiver.

22.4.1 Additional attribute types of Transportable

TransportablePacking

Term
TransportablePacking

Definition
An enumerated type of packing where goods could be packed in.

Description
When transporting goods they are often packed in a kind of packing. This enumeration type gives common kinds of packing.

Data structure

Packing	Description
UNDEFINED	Packaging unknown
ENVELOPE	Postage envelope
PARCEL	A parcel, usually packaging for several ordered items
ISOPALLET	A pallet with goods, the pallet has dimensions 100x120x12 cm
EURPALLET	A pallet with goods, the pallet has dimensions 80x120x12 cm
BAG	A bag with goods, for example a postbag
BARREL	A barrel, for example with fluid
TUBE	A packaging that has a tubular shape
ISOCONTAINER	A shipping container based on ISO container dimensions
BOX	A packaging that is bigger than a parcel
RAW	Raw materials, for example coal or grit

Route Location

Term
RouteLocation

Definition
A location on a route where a transportation company has to load or unload goods.

Description
A transportation company visits one or more location on a route to load or unload goods. Loading and

unloading are referred to as a task that has to be performed at the location. The RouteLocation defines where the goods have to be loaded or unloaded.

Every RouteLocation has a a list of time windows when the task is preferred to be executed. The RouteLocation also has a time window when the task is scheduled by the transportation company to be executed. Finally, the RouteLocation has a time window then the task is actually executed.

It is possible to add a instruction to the RouteLocation gives more specific information about the execution of the task. If the task is executed then a POP (loading) or POD (unloading) is defined.

Data structure

Attribute	Type	Req	Default	Description
Company	Client	M	N/A	Information about the company where the operation has to be performed
TaskOperation	TaskOperation	M	N/A	Indicated the operation that needs to be performed
TaskRequested	List of TimeWindow	A	NULL	Time windows that tell when the operation is requested to be performed
TaskEstimated	TimeWindow	O	NULL	The time window when the operation is expected
TaskPlanned	TimeWindow	O	NULL	The time window when the operation is planned.
TaskActual	TimeWindow	O	NULL	The time window when the operation was done
Instruction	String[1024]	O	“”	Extra instructions for the operation
Reference	String[100]	O	“”	A reference that is needed at the location of Company.
POE	ProofOfExecution	O	NULL	POP or POD of the task. The value of TaskOperation defines whether the value of POE is a POP or a POD.

Transportable Dimension

Term
TransportableDimension

Definition
Combination of values, that can be abbreviated by an enumerated attribute, for several units which combined give information about the dimension of a transportable.

Description
A TransportableDimension gives information about the size of a transportable. It is possible to use standard sizes as well as custom sizes. If a standard size is used then the Measurement, Height, Width, Depth, Diameter, Liquid and Cubic attribute can be ignored. If a custom size is given using TransportableDimension then the value of enumerated type Measurement indicates which other attributes are required.

Data structure

Attribute	Type	Req	Default	Description
StandardSize	StandardSize	C	NULL	If a transportable has a standard size then it is not needed to specify any dimensions. If it hasn't a standard size then it needs to have dimensions.
Measurement	Measurement	C	NULL	Kind of measurement (cubic, etc). If a StandardSize is given then this attribute is ignored.
Height	Float	C	0	Height of the transportable in meters. If a StandardSize is given then this attribute is ignored. This attribute is mandatory if the value of Measurement indicates this.
Width	Float	C	0	Width of the transportable in meters. If a StandardSize is given then this attribute is ignored. This attribute is mandatory if the value of Measurement indicates this.
Depth	Float	C	0	Depth of the transportable in meters. If a StandardSize is given then this attribute is ignored. This attribute is mandatory if the value of Measurement indicates this.
Diameter	Float	C	0	Diameter of the transportable in meters. If a StandardSize is given then this attribute is ignored. This attribute is mandatory if the value of Measurement indicates this.
Liquid	Float	C	0	Amount of liquid of the transportable in liters. If a StandardSize is given then this attribute is ignored. This attribute is mandatory if the value of Measurement indicates this.
Cubic	Float	C	0	Cubic size of the transportable in square meters. If a StandardSize is given then this attribute is ignored. This attribute is mandatory if the value of Measurement indicates this.
Weight	Float	O	0	Weight in kilogram
FixedOrientation	Boolean	O	0	If the transportable may only be rotated around the vertical axis then it has a fixed orientation and should the value be true. A glass of water is a typical transportable with a fixed orientation.

ADR Information

Term
ADRInformation

Definition
European agreement for the identification of dangerous goods which are transported.

Description
ADR is an abbreviation for 'Accord européen relatif au transport international des marchandises Dangereuses par Route' what is an identification system for dangerous goods. An ADR sign consists of two attributes, namely the Kemler code and the Identification Number of the good given by the United Nations.

Data structure

Attribute	Type	Req	Default	Description
KemlerCode	String[4]	M	N/A	The Kemler code of the ADR specification
UNNr	String[4]	M	N/A	The UN number of the ADR specification

Range of temperature

Term
TemperatureRange

Definition
A temperature range that consists of a minimum and a maximum temperature.

Description
A temperature range that consists of a minimum and a maximum temperature. If only a single temperature is meant then the minimum temperature and the maximum temperature are equal. The minimum temperature can never exceed the maximum temperature.

Data structure

Attribute	Type	Req	Default	Description
TemperatureMinimum	Int	M	N/A	Minimum temperature of the range in degree Celcius
TemperatureMaximum	Int	M	N/A	Maximum temperature of the range in degree Celcius

22.4.2 Additional attribute types of Transportable/RouteLocation

TaskOperation

Term
TaskOperation

Definition
Enumerated type that defines the operation of a task.

Description
Enumerated type that defines the operation of a task.

Data structure

TaskOperation	Description
UNKNOWN	Unknown task
LOAD	Goods have to be loaded
UNLOAD	Goods have to be unloaded

Time Window

Term
TimeWindow

Definition
A time window that is defined by a starting and ending point.

Description
A TimeWindow describes a period in time. The ending point has to be the same or later in time than the starting point. The time between the starting and ending point is the time window. If a single point in time is meant then the starting and ending point are equal.

Data structure

Attribute	Type	Req	Default	Description
TimeBegin	ISO-8601 (date and time)	M	N/A	Begin of the time window
TimeEnd	ISO-8601 (date and time)	M	N/A	End of the time window

Proof Of Execution

Term
ProofOfExecution

Definition
Proof Of Execution is a proof of pick-up or delivery of goods.

Description
The Proof Of Execution has a name, moment in time and a signature of the natural person that approved the pick-up or delivery.

Data structure

Attribute	Type	Req	Default	Description
Name	String[50]	M	N/A	Name of the person who signed for the execution
Moment	ISO-8601 (date and time)	M	N/A	Moment of execution
Signature	String[8192]	M	N/A	Signature of execution, format specification yet unknown

Standard sizes

Term
StandardSize

Definition
An enumerated type of sizes of transportables which are commonly known and usually recognized by a standard organization.

Description
A size of a transportable that is commonly known and usually recognized by a standard organization such as ISO. Typical standard sizes are sea containers and pallets.

Data structure

StandardSize	Description
ISOTAINER20H	Shipping container of 20 ft length and 9 ft 6 height
ISOTAINER40H	Shipping container of 40 ft length and 9 ft 6 height
ISOTAINER45H	Shipping container of 45 ft length and 9 ft 6 height
ISOTAINER48H	Shipping container of 48 ft length and 9 ft 6 height
ISOTAINER53H	Shipping container of 53 ft length and 9 ft 6 height
ISOTAINER20L	Shipping container of 20 ft length and 4 ft 3 height
ISOTAINER40L	Shipping container of 40 ft length and 4 ft 3 height
ISOTAINER45L	Shipping container of 45 ft length and 4 ft 3 height
ISOTAINER48L	Shipping container of 48 ft length and 4 ft 3 height
ISOTAINER53L	Shipping container of 53 ft length and 4 ft 3 height
ISOPALLET24	ISO Pallet (100x120x12 cm) with a maximum height of 24 inches including pallet height
ISOPALLET36	ISO Pallet (100x120x12 cm) with a maximum height of 36 inches including pallet height
ISOPALLET50	ISO Pallet (100x120x12 cm) with a maximum height of 50 inches including pallet height
ISOPALLET70	ISO Pallet (100x120x12 cm) with a maximum height of 70 inches including pallet height
ISOPALLET72	ISO Pallet (100x120x12 cm) with a maximum height of 72 inches including pallet height
ISOPALLET77	ISO Pallet (100x120x12 cm) with a maximum height of 77 inches including pallet height
ISOPALLET79	ISO Pallet (100x120x12 cm) with a maximum height of 79 inches including pallet height
EURPALLET60	EUR Pallet (80x120x12 cm) with a maximum height of 60 centimetre including pallet height
EURPALLET95	EUR Pallet (80x120x12 cm) with a maximum height of 95 centimetre including pallet height
EURPALLET125	EUR Pallet (80x120x12 cm) with a maximum height of 125 centimetre including pallet height
EURPALLET178	EUR Pallet (80x120x12 cm) with a maximum height of 178 centimetre including pallet height
EURPALLET185	EUR Pallet (80x120x12 cm) with a maximum height of 185 centimetre including pallet height
EURPALLET190	EUR Pallet (80x120x12 cm) with a maximum height of 190 centimetre including pallet height
EURPALLET200	EUR Pallet (80x120x12 cm) with a maximum height of 200 centimetre including pallet height
ULD2C	Unit Load Device (aircraft) container of 120 ft^3 / 3.4 m^3
ULD3C	Unit Load Device (aircraft) container of 153 ft^3 / 4.3 m^3

StandardSize	Description
ULD6C	Unit Load Device (aircraft) container of 316 ft^3 / 8.8 m^3
ULD8C	Unit Load Device (aircraft) container of 243 ft^3 / 6.9 m^3
ULD11C	Unit Load Device (aircraft) container of 253 ft^3 / 7.2 m^3
ULD8P	Unit Load Device (aircraft) pallet of 243 ft^3 / 6.9 m^3
ULD11P	Unit Load Device (aircraft) pallet of 253 ft^3 / 7.2 m^3

Measurement

Term
Measurement

Definition
Measurement is an enumerated type that indicates which values are required to define a specific dimension.

Description
There are different kind of dimensions such as cubic meters for coals and the combination of diameter and height for a coil. Measurement indicates which kind of dimension is given and what attributes are required for that kind of dimension.

Data structure

Measurement	Description	Required size attributes
CUBIC	The volume is given by the size	Cubic
HWD	The Height, Width and Depth are given by the size	Height, Width Depth
DH	The Diameter and Height are given by the size	Diameter, Height
LIQUID	The amount of fluid is given by the size	Liquid

22.5 TransportableTrack entity

Term
TransportableTrack

Definition
A (part of a) route of a transportable that has no influence on the original location or final destination.

Description
The data structure of a transportable practically only contains information about the origin, destination and history. To give information about a part of the track of the transportable it is possible use TransportableTrack. If a transport company only needs to do a part of whole transport track then a TransportableTrack can be added to a transportable. This information cannot be included in Transportable, because a track differs for every transport company involved in the whole transportation process while the information about the transportable is equal.

Data structure

Attribute	Type	Req	Default	Description
TransportableID	UUID	M	N/A	Identifier of the transportable where this track is attached to.
LocationFrom	RouteLocation	O	NULL	Location where the transportable needs to be picked up if differtent from LocationOrigin
LocationNext	RouteLocation	O	NULL	Location where the transportable needs to be delivered if different from LocationFinal
LocationReturn	RouteLocation	O	NULL	Location where the transportable needs to be returned if different from LocationReturn of the transportable.

22.6 LocationMoment entity

Term
LocationMoment

Definition
A LocationMoment defines a moment that a transportable was at a specific location.

Description
The LocationMoment data structure is used to describe a specific location with a reference to a transportable. This defines the moment that the transportable was physically at that location at that moment. Optionally a reference to a holder can be given to be able to provide more information about the location.

Data structure

Attribute	Type	Req	Default	Description
TransportableID	UUID	M	N/A	Identifier of the transportable where this LocationMoment refers to.
Location	Location	M	N/A	Geographic coordinates or address to uniquely identify the location on earth. The value of this attribute indicates where the transportable was located.
Moment	ISO-8601 (date and time)	M	N/A	Moment that the transportable was at the location
HolderID	UUID	O	NULL	Holder entity that can provide more information about the location.

22.6.1 Additional attribute types of LocationMoment

Location

Term
Location

Definition
A location consists of a GeoLocation, an Address or both with the purpose to uniquely identify a fixed location on earth.

Description
Both an address and the combination of longitude and latitude can identify a fixed location on earth. Sometimes it could not be known what the address is, but it isn't known what the coordinates are. The opposite could also occur. It is sure that a Location attribute will uniquely identify a location on earth although it is not sure whether this is done by a coordinate and/or a address. If both an address and coordinates are given they have to identify the same location on earth.

Data structure

Attribute	Type	Req	Default	Description
Address	Address	C	NULL	An address that identifies a location. Either the Address attribute or the GeoLocation attribute needs to be given. Both are only allowed if they identify the same location.
GeoLocation	GeoLocation	C	NULL	A coordinate on earth that identifies a location. Either the Address attribute or the GeoLocation attribute needs to be given. Both are only allowed if they identify the same location.

GeoLocation

Term
GeoLocation

Definition
A GeoLocation is the location on earth in decimal degrees based on WGS84.

Description
A GeoLocation consists of two floating point numbers. These numbers tell the latitude and the longitude in decimal degrees.

Data structure

Attribute	Type	Req	Default	Description
Latitude	Float	M	N/A	Latitude coordinate North
Longitude	Float	M	N/A	Longitude coordinate East

22.7 Holder entity

Term
Holder

Definition
A holder is a resource that contains one or more transportables or other holders.

Description
A holder has a fixed or variable location and contains transportables or other holders. Typical examples of holders are warehouses, vehicles, persons, shipping containers and pallets. All these holders contain transportables. A holder is a long-lived resource.

Every holder can be contained by another holder. Typical example of this is a pallet that is contained by a vehicle. The location of the vehicle is therefore the location of the pallet.

Data structure

Attribute	Type	Req	Default	Description
ID	UUID	M	N/A	Unique identifier of the holder
Category	HolderCategory	M	OTHER	Category of the holder
Subcategory	HolderSubcategory	M	OTHER	Subcategory of the holder
Description	String[150]	O	“”	A description of the holder.
FixedLocation	Boolean	M	N/A	True if a holder is always at the same location (like a building). False if a holder can move (like a car).
Location	Location	M	NULL	Geographic coordinates or address to uniquely identify the location on earth. The value of this attribute indicates where the holder is located.

22.7.1 Additional attribute types of Holder

Holder Category

Term
HolderCatergory

Definition
An enumerated type that gives the category of a holder.

Description
An enumerated type that gives the category of a holder.

Data structure

Categorie	Description
OTHER	Not defined or unknown category
BUILDING	The holder is a building like a warehouse or a storage facility
PERSON	A natural person
ROADTRANSPORT	Means of transport that drives on roads like trucks.
WATERTRANSPORT	Means of transport that goes over water like boats.
AIRTRANSPORT	Means of transport that goes through the air like an aircraft.
RAILTRANSPORT	Means of transport that drives on a track like trains.
ENCAPSULATION	The holder is used for encapsulation of other holders/transportables like shipping containers.
SPECIAL	Special predefined holder in the transportation proces

Holder Subcategory

Term
HolderSubcategory

Definition
An enumerated type that identifies a subcategory of a category of a holder.

Description
A holder belongs to a category and a subcategory. Categories are more general than subcategories.

Data structure

Subcategory	Category	Description
OTHER	OTHER	A subcategory that is not defined or unknown
STORAGEROOM	BUILDING	'Small' size part of a building used for storage
DEPOT	BUILDING	'Medium' size building or large part of a building for storage
WAREHOUSE	BUILDING	'Big' size building completely dedicated to storage of goods
PERSON	PERSON	A human person that can carry a transportable, for example a mailman
CAR	ROADTRANSPORT	A passenger car used for transport
SMALLCARGOVAN	ROADTRANSPORT	A small cargo van where the cargo space is not separated from the driver for example a Volkswagen Caddy
CARGOVAN	ROADTRANSPORT	A cargo van with a bigger cargo space that is usually separated from the driver, for example a Mercedes Vito or a Ford E350
BIGCARGOVAN	ROADTRANSPORT	A cargo van that is longer and higher than a cargo van, for example a Mercedes Sprinter or Volkwagen LT
CUBEVAN	ROADTRANSPORT	A van with a specially build up cargo space that is cube-shaped
STEPVAN	ROADTRANSPORT	A special cargo van with the size of a cargo van or a big cargo van that has a special door that makes it easy to step in and out of the car
STRAIGHTTRUCK	ROADTRANSPORT	A truck where the cabin and the cargo space can not be separated
TRAILER	ROADTRANSPORT	A trailer that is part of a tractor trailer combination
PICKUPTRUCK	ROADTRANSPORT	A truck with a separate cabin en cargo bed of few square meters. The cargo bed is surrounded by short rigid sides and an opening rear gate.
BIKE	ROADTRANSPORT	A cycle
MOTORCYCLE	ROADTRANSPORT	A motorcycle
GONDOLACAR	RAILTRANSPORT	An open railroad car for raw materials
BOXCAR	RAILTRANSPORT	A closed railroad car that carries general freight
FLATCAR	RAILTRANSPORT	A flat and open railroad car for example for transporting wood or other freight that would not fit in a flatcar. These cars are also used for transporting sea containers.

Subcategory	Category	Description
HOPPERCAR	RAILTRANSPORT	An open railroad car like a gondola car with special openings at the bottom or sides to load/unload
TANKCAR	RAILTRANSPORT	A railroad car to transport liquefied loads
STOCKCAR	RAILTRANSPORT	A railroad car to transport livestock
CONTAINERWELLCAR	RAILTRANSPORT	A railroad car made for the transport of sea containers. The container is placed in a gondola that is very close to the rail. This lowers the point of gravity and enabled double-stacking of containers.
COILCAR	RAILTRANSPORT	A railroad car specialized for coils like steel coils
ROADRAILERCAR	RAILTRANSPORT	A railroad car specialized for loading trailers of tractor trailer combinations
CARGOPLANE	AIRTRANSPORT	A specialized plane for transporting cargo
CIVILPLANE	AIRTRANSPORT	A plane that is used for civilian as well as cargo transports
BARGE	WATERTRANSPORT	A cargo ship that goes inland
BULKCARRIER	WATERTRANSPORT	A cargo sea ship
CONTAINERSHIP	WATERTRANSPORT	A cargo sea ship that is specially made for the transportation of sea containers
TANKER	WATERTRANSPORT	A sea ship that is specially made for the transport of fluids, most likely oil
ISOCONTAINER	ENCAPSULATION	A shipping container based on ISO container dimensions
ORIGIN	SPECIAL	The origin where the goods are coming from
DESTINATION	SPECIAL	The final destination of the goods
LOST	SPECIAL	The goods are lost
LOSTANDTRACKING	SPECIAL	The goods are lost but at this moment there is an active tracking process to find the goods.

23 Bibliography

[Alpern et al.] Recognizing safety and liveness, Bowen Alpern, Fred B. Schneider, Cornell University; 1986

[ANSIX12] American National Standards Institute ASC X12 standard; http://www.x12.org

[BPMN] Business Process Modeling Notation, http://www.bpmn.org/

[Borysowich] Prototyping: Types of Prototypes; Craig Borysowich; Imagination Edge Inc; 2007; http://it.toolbox.com/blogs/enterprise-solutions/prototyping-types-of-prototypes-14927

[Caldwell] Graph Theory Glossery; Dr. Chris K. Caldwall; University of Tennessee; 1995

[Camarinha] Distributed database overview; Prof. L.M. Camarinha-Matos; UNINOVA – New University of Lisbon; http://www.uninova.pt/~cam/is/ddb.doc

[CourierExchange] Transport Exchange Group LTD; 80 Scrubs Lane; London; UK; http://www.courierexchange.co.uk/

[CoverPages-RN] RosettaNet Technology Report; Robin Cover; Cover Pages and OASIS; 2004; http://xml.coverpages.org/rosettaNet.html

[Dahlin et al.] Data Synchronization for Distributed Simulations, Mike Dahlin, Aslan Brooke, Muralidhar Narasimhan, Bruce Porter, University of Texas; 2001

[DHL-EDI] DHL Logbook; Electronic Data Interchange; http://www.dhl-discoverlogistics.com/cms/en/course/technologies/connection/edi.jsp

[Doyle] Foundations of Computer Networking; Dr. John F Doyle; Indiana University Southeast; fall 2007; B438 Syllabus

[ebXML] Electronic business Extensible Markup Language; Suit of specifications to conduct business over the Internet; http://www.ebxml.org

[ebXML-RS] ebXML Requirement Specification version 1.0.6; Michael C. Rawlins et al.; UN/CEFACT and OASIS; 2001

[ebXML-TAS] ebXML Technical Architecture Specification version 1.0.4; Anders Grangeard et al.; UN/CEFACT and OASIS; 2001

[EDIFACT] Electronic Data Interchange for Administration, Commerce and Transport; UNECE; http://www.unece.org/trade/untdid/welcome.htm

[EDIFICE] European User Group for the Electronics Industry; European RosettaNet User Group; http://www.edifice.org/

[ELPIF-patent] United States Patent 20030191677 ; filed March 27th 2002; published October 9th 2003; http://www.freepatentsonline.com/y2003/0191677.html

[freebxml] freebXML, a royalty-free open source ebXML registry project; http://ebxmlrr.sourceforge.net/

[Gray et al.] The dangers of Replication and a Solution, Jim Gray, Pat Helland, Microsoft; Patrick O'Neil, UMB; Dennis Shasha, NYU; 1996

[Greenwald et al.] Agreeing to Agree: Conflict Resolution for Optimistically Replicated Data, Michael B. Greenwald, Sanjeev Khanna, Keshav Kunal, Benjamin C. Pierce, Alan Schmitt, DISC; 2006

[Intellicom] Intellicom B.V.; Aalsmeerweg 79; Amsterdam; NL; http://www.intellicom.nl/

[ISO] International Organization for Standardization; Located in Geneva Switzerland; http://www.iso.org/iso/search.htm?qt=&searchSubmit=Search

[ke7] Distributed Processing and Distributed Databases; King Edward VII School; United Kingdom http://learningat.ke7.org.uk/itweb/year13/distproc2.htm

[Koolwaaij] XML Hype of Hoop?; Johan Koolwaaij; Telematica Institute; 2000; https://doc.telin.nl/dsweb/Get/File-10901/XML%20hype%20of%20hoop%20(Ned).ppt

[Linz] An Introduction to Formal Languages and Automata, 2nd edition, Peter Linz, University of California; 1997

[MVC-pattern] Model-View-Controller design pattern; Java Blueprints; Sun Microsystems; http://java.sun.com/blueprints/patterns/MVC-detailed.html

[OASIS] Organization for the Advancement of Structured Information Standards; http://www.oasis-open.org

[OMG] Object Management Group, http://www.omg.org/

[papiNet] Paper Industry Network providing a standard for automation between these companies; http://www.papinet.org

[papiNet-v230] Paper Industry Network messages for electronic communication version 2.30; http://www.papinet.org/index.php?id=101

[PCMagGateway] PC Magazine, the independent guide to technology; http://www.pcmag.com/encyclopedia_term/0,2542,t=gateway&i=43670,00.asp ; spring 2008

[Reenskaug] The original MVC reports; Trygve Reenskaug; University of Oslo; 1979; http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf

[RootDNS] Internet Root nameservers; http://en.wikipedia.org/wiki/Root_nameserver

[RosettaNet] RosettaNet; http://www.rosettanet.org/

[RosettaNet-CSP] RosettaNet Overview of Clusters, Segments and PIPs version 02.04.00; RosettaNet Program Office; 2008; http://portal.rosettanet.org/cms/export/sites/default/RosettaNet/Downloads/RStandards/ClustersSegmentsPIPsOverview_23April2008.pdf

[RosettaStone] The Rosetta Stone; http://en.wikipedia.org/wiki/Rosetta_Stone

[Sadoski] Three Tier Software Architecture; D. Sadoski and S. Comella-Dorda; Carnegie Mellon University SEI; 2000; http://www.sei.cmu.edu/str/descriptions/threetier.html

[Silberschatz] Database System Concepts, A. Silberschatz, H. Korth, Bell Laboratories; S.

Sudershan, IIT; 3rd Edition 1997

[ShiwaFu] A Practical Approach to Web-Based Internet EDI; Shiwa Fu et al.; IBM T.J. Watson Research Center; 1999

[StylusStudio] EDIFACT Sample Converted to XML; http://www.stylusstudio.com/EDI/EDIFACT_example.html

[UDDI] OASIS Standard for Universal Description Discovery and Integration; http://uddi.xml.org

[UMM] UN/CEFACT Modeling Methodology; http://www.unece.org/cefact/umm/umm_index.htm

[Wiesmann et al.] Understanding Replication in Databases and Distributed Systems; M. Wiesmann, F. Pedone, A. Schiper, EPFL; B. Kemme, G. Alonso, ETHZ; 2000

[Tanenbaum] Computernetwerken; A. S. Tanenbaum; 2nd edition 1997; ISBN 9039505578

[TanenbaumProxy] Computernetwerken; A. S. Tanenbaum; 2nd edition 1997; ISBN 9039505578; Chapter 7: The application layer

[Tedim-LDI] Advantages of XML/EDI in logistics data interchange; P. Malmi, A. Lantonen; TEDIM; Ministry of Transport and Communication of Finland; 1999; http://www.tedim.com/default.asp?file=821

[TLNNOV06] Leveranciers transport management software http://www.tln.nl/media/Consultancy/Leveranciers/Transportmanagement_systemen.pdf

[WikiDD] Wikipedia, Distributed Database, Summer 2008 http://en.wikipedia.org/wiki/Distributed_database

[WikiGateway] Wikipedia, http://en.wikipedia.org, subject 'Gateway (telecommunications)'; spring 2008

[WikiProxy] Wikipedia, http://en.wikipedia.org, subject 'Proxy server'; spring 2008

[Zhang] ELPIF: An E-Logistics Processes Integration Framework Based in Web Services; Liang-Jie Zhang et al.; IBM T.J. Watson Research Center; 2001