1 SEPTEMBER 2018

INTEGRATING A PLATFORM FOR EARLY PAYMENT WITH DIFFERENT ERP SYSTEMS

TIM KRAAI
CAPE GROEP

CAPEGROEP.NL
grensverleggend vooruit

UNIVERSITY OF TWENTE.

Publication date: 6 September 2018

Student

T. Kraai (Tim)
Bachelor student Industrial Engineering and Management
University of Twente
Student number: s1724150

Supervisors

1st supervisor University of Twente
Prof. Dr. M.E. Iacob (Maria)
Professor

2nd supervisor University of Twente
PhD. L.O. Meertens (Lucas)
Assistant Professor

Supervisor from CAPE Groep
T. ten Vregelaar (Tom)
Account Manager

Management Summary

This research helps to eliminates the manual tasks in the data exchange between Enterprise Resource Planning (ERP) systems and the platform for early payment, by providing a Canonical Data Model (CDM) that reduces the complexity of the data mapping used to create an automated data exchange.

Company A provides a platform for early payment where a discount on invoices can be negotiated in exchange for earlier payment. In the current situation, the buyer must upload the invoices for which he wants to receive a discount manually to the platform. When a discount is negotiated, the results must be entered manually into the ERP system of the buyer. This is a time intensive process and discourages the buyer to use the platform for early payment.

To eliminate the manual tasks during data exchange and create an automated process, an integration is needed. There are multiple buyers with different ERP systems, which each have a unique data structure. To integrate these systems, the Enterprise Application Integration type is used to minimize the number of integrations that must be created. The Enterprise Service Bus uses the principles of Enterprise Application Integration and creates a common understanding of the data structures that are integrated with a Canonical Data Model. The advantage of this CDM is that it is a system-independent generic model that is understandable for every system. A disadvantage is that no structured methods for the development were identified, this results in models that are created in an intuitively ad-hoc way in the creator's mind.

To validate the use of a CDM for the automation of the data exchange a prototype was created. This prototype is an Enterprise Service Bus that contains a CDM which is developed based on the GS1 standard for invoices to create the system-independent generic model.

In the current process, the time needed to manually upload an invoice to the platform for early payment was on average 3:19 minutes. The prototype of the automated process that was created with the CDM performed this upload in about one second without any user involvement.

Currently, the created prototype supports the invoice upload, the entering of the results from the early payment process in the ERP system of the buyers still must be developed. Because the creation of a CDM is difficult without a structured method or language for semantic understanding, an industry independent standard is useful for support during the development. Future research to identify a structured method would decrease the time needed for the development of a CDM and make the development less intuitive.

Preface

This is the report of my bachelor project at CAPE Groep about the integration of a platform for early payment with different ERP systems. This project is the completion of my bachelor Industrial Engineering and Management at the University of Twente.

During a period of six months, I worked at CAPE Groep on my project. I would like to thank CAPE Groep for the opportunity to work on this project. During the project, I learned a lot about the dynamic discounting, invoice process and the development of integrations in eMagiz.

I would like to thank Tom ten Vregelaar for his time and supervision during the project. When I had questions or wanted new input he was able to help me and gave me new challenges.

I would also like to thank Maria Iacob and Lucas Meertens for the support during the project and suggestions of topics that could be interesting for my research. Special thanks to my parents for supporting me in the writing process and helping me to keep going.

I hope this project will help the integration of all the future buyers to the platform for early payment of company A.

Tim Kraai
August 2018

Table of Contents

1. Introduction

In this chapter first CAPE Groep and Company A will be introduced. The problem that was identified as the basis for this project is described in 1.1. In 1.2 the relevance of this problem is discussed. Based on the problem the goal of this project is described in 1.3 and the main and sub research questions are defined in 1.4 and 1.5. In 1.6 the scope of this project is defined and 1.7 provides an outline of the thesis.

About CAPE Groep

CAPE Groep is a consultancy company that specializes in integrating IT-solutions. By working together with the customer, they aim to implement innovative IT-systems that reduce costs and make the company flexible to adopt new technologies.

The IT-solutions that CAPE Groep delivers to the customers are built in Mendix and eMagiz. Mendix is an application Platform as a Service (aPaaS). With this platform, applications can be developed that can be quickly integrated and used by the customers. eMagiz is an integration Platform as a Service (iPaaS) that is used to integrate the existing application of the customer with Mendix and other applications.

Company A

Company A is a FinTech company that aims to reduce the amounts of cash that is stuck in the supply chain due to long payment terms. To achieve this, they build a platform where buyers can upload an invoice and offer to pay the invoice before the due date for a certain discount percentage. When the supplier accepts the discount, the buyer can start the payment. Company A receives a small fee for facilitating the early payment.

In an innovation project from the department of entrepreneurship from the Ministry of Economic Affairs and Climate Policy of the Netherlands, CAPE Groep and Company A received a subsidy to create an automated link between the ERP systems from the customers of Company A to the platform for early payment of Company A.

1.1 The problem

When a customer buys a product from a supplier, the customer receives the product and an invoice. This invoice contains the payment terms, invoice amounts, and optional discounts. The payment terms include the period for the payment, the condition for that payment and the discounts that may be received. The period for payment is on average 24 days in the Netherlands (Atradius, 2017). This means that suppliers must wait 24 days on average for the payment of their deliveries.

For the supplier it is expensive to take out a short-term loan to cover the gap between the delivery of the goods or services and the time the invoice is paid. To reduce the period to payment, suppliers can offer a discount period on their invoices. This means that if their invoice is paid during this period, a small discount is applicable. This gives the buyer a way to use his extra cash as an investment and lets the supplier receive the payment earlier. But because the discount period is static, it is still more lucrative for the buyer to pay at the end of this period. In some cases, a buyer has enough cash but lacks a way to invest his money. For instance: the period that they have excess cash is too short to put it on their bank account and retrieve a considerable amount of interest.

For the supplier to receive payment earlier, dynamic discounting is introduced. It lets suppliers daily receive earlier payment of the invoice in exchange for a discount on the invoice, based on the number of days the invoice is paid before the due date (Gelsomino, Mangiaracina, Perego, & Tumino, 2016). To facilitate this dynamic early payment of invoices Company A provides a platform for early payment, which can be used to negotiate the discount with the supplier. To negotiate a discount for an invoice, the buyer must upload the invoice to the platform for early payment together with a proposed payment date and a discount rate. In the current situation, buyers must upload their invoices manually to the platform of company A, by importing a file with the correct format and data into the platform for early payment. This manual action takes time for the buyer this time is costly for the supplier and the buyer. But both the buyer and the supplier must wait until the file of the buyer is imported into the platform for early payment before the supplier can respond to the proposal of the buyer. In this way, the platform for early payment becomes an obstacle in the negotiation between buyer and supplier and can result in a later payment to the supplier and a lower discount for the buyer.

For company A the uploading of an invoice must be as easy as possible because they receive a small commission per invoice. The more invoices are paid earlier using the platform for early payment, the higher their profit.

1.2 The relevance of the problem

For the supplier, early payment leads to a better working capital and a smaller gap between their shipment of the product to the customer and the time they receive their money. When their working capital is low, it is better to get paid earlier with a small discount than taking a loan from the bank.

The buyer benefits from the earlier payment because excess cash can be used to receive a good return by paying the invoice as early as possible. It also helps to get a better relationship with the supplier, because they can rely on the fact that their invoices are paid

before the final due date and don't have to send reminders.
When the discount and the terms are negotiated, the results must be entered into the financing system of the buyer. This is often an ERP system, (see Explanation of Terms in Appendix C for clarification). Because most ERP systems don’t support a discount per days, the entering of this discount is a complex process that requires expertise in the ERP software.

For Company A the commission is a percentage of the invoice. Every invoice may be entered in the platform for early payment. The number of invoices that are paid earlier using the platform for early payment is the main source of profit of Company A.
To increase the number of invoices, it must be easy for buyers and suppliers to start negotiating early payment for an invoice. Also, the number of buyers that are connected to the platform must increase to receive more invoices. This means the obstacle of entering invoices manually into the platform for early payment must be solved, this can be done by using IT. The buyers will profit in saved time on entering invoices, can propose earlier payment dates and therefore higher discounts and receive quicker responses from the supplier. Also, Company A will receive more commission.

For society earlier payment of invoices would mean that the cash flows quicker between companies because the companies receive the invoice amount before the end of the payment period. Society would benefit from this as a higher liquidity leads to more investments because there is less cash stuck in the supply chain.

1.3 The goal of the project

The first goal of this project is to eliminate the manual tasks that must be performed, on the one hand when uploading an invoice to the platform for early payment and on the other hand when entering the agreed terms and discounts in the ERP system of the buyer.

The second goal of this project is to create a Canonical Data Model (CDM) to decrease the time needed to exchange data between the different ERP systems of buyers and the platform for early payment to support the automation of this exchange. This automation requires data integration of the ERP systems and the platform for early payment of Company A.

Literature research will be performed on a CDM to integrate these systems. This research can be used to create a CDM in a prototype to upload invoices from one ERP-system to the platform for early payment of company A. This prototype will serve as a proof of concept. Based on this project, recommendations for the other connections will be given.

1.4 Research question

During the project, different approaches to reach the project goal will be researched. This research will then be used in the project and the prototype. The research goal can be translated into the following research question:

What is a suitable CDM to reduce the complexity of data mapping in connecting a platform for early payment, that facilitates dynamic discounting, with different ERP systems?

The focus of this research is the scalability and changeability of a connection between a platform for early payment and multiple ERP systems. A CDM to integrate systems will be researched in literature, and a suitable method of integration will be chosen for the multiple systems that are involved.

1.5 Research methodology and research questions

There are several methodologies to manage a research project. For this project, the Design Science Research Methodology (DSRM) was chosen because this methodology uses design science which is aimed at creating and evaluating IT artifacts to solve problems in organizations. DSRM is used in this project because a prototype, an artifact, will be created to solve and validate the solution for this project. In DSRM, a framework is provided to perform design science research for information systems (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). The DSRM consists of six phases which are described below and displayed in Figure 1.

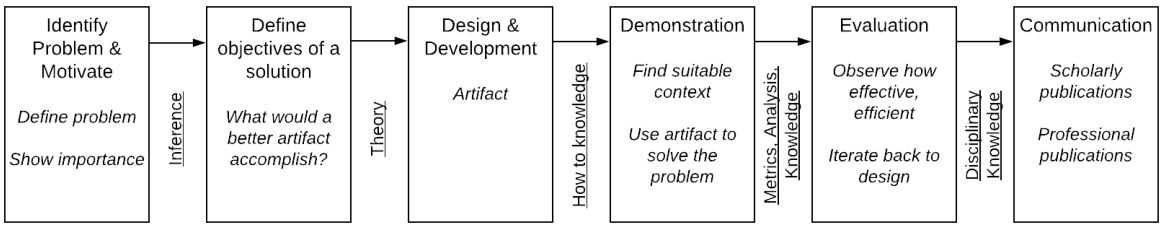


Figure 1 DSRM Process Model

The first phase of the DSRM is the problem identification. In this phase, the current process will be described and the problems that are experienced. To clearly define requirements for the solution, it is important to describe the current process in detail and show the importance of the problem. The following questions will be answered in this phase:

1. *What does the manual early payment process look like?*
   - *1.1. What is dynamic discounting?*
   - *1.2. What are the phases in the dynamic discounting process?*
   - *1.3. How are new invoices added to the platform for early payment?*
   - *1.4. What are the problems with the manual early payment process?*
   - *1.5. How are the early payment results send to the buyer?*

In the second phase of the DSRM, the objectives and solutions are defined. Information for the first and second phase will be gathered from employees of CAPE Groep and Company A.

2. *What does the ideal automated process look like?*
   - *2.1. Which process steps are to be automated?*
   - *2.2. What are the problems in integrating the different ERP systems?*

From the goals and requirements, the conclusion was drawn that the automation is extensive and requires integration. To determine the best type for this integration a research will be performed with the research question:

3. *What is a suitable integration type for integrating multiple different ERP systems with a platform for early payment?*

For the development of an Enterprise Service Bus solution, a Canonical Data Model must be developed, the following research question is used to determine how a CDM is created:

4. *How to design and develop a CDM?*

During the design & development phase, a prototype will be developed for one ERP system. In the fourth phase of the DSRM, the prototype will be demonstrated.

5. *Is it possible to create a working prototype with a CDM?*
   - *5.1. How is the prototype developed?*
   - *5.2. What does the CDM look like?*
   - *5.3. Does the developed prototype reduce the time needed for the data exchange?*

During the evaluation, the prototype and solution will be evaluated based on the requirements that were defined and the prototype. The following research question will be used for this:

6. *Does the introduction of an ESB with a CDM solve the project problems?*
   - *6.1. How are the problems solved?*
   - 6.2. *Is the solution also future proof?*

In the last communication phase, recommendations will be given for improvements based on the results and possible follow-up research will be identified. The questions that will be answered in this phase are:

7. *What can be concluded from this project?*
   - *7.1. Is the project goal reached?*
   - *7.2. What are the recommendations?*
   - *7.3. What could be further researched?*

1.6 The scope of the project

In this project the focus will be on the exchange of data between the ERP system of the buyer and the platform for early payment. The project will not analyze or change the administrative process that is performed within these systems and the platform.

The selection process of invoices in the ERP system that are to be uploaded to the early payment will not be discussed, because it is outside the scope of this project. Also, the payment process that is performed in the ERP system of the buyer will not be researched and implemented in this project and prototype.

For all the invoices and discounts that are agreed on the platform, the assumption is made that the amounts are in euros. The problem is that exchange rates change constantly and will influence the amount to be paid. If other currencies are to be added to the system, agreements must be made about the moment exchange rates are to be settled.

During this project, a solution will be developed for connecting multiple ERP systems to the platform for early payment. The proof of concept in the prototype, however, will focus on connecting just one ERP system to the platform for early payment.

1.7 Thesis structure

Chapter	DSRM	Research questions
2. Current process	***Phase 1 Identify problem & Motivate***	*1. What does the manual early payment process look like?* *1.1. What is dynamic discounting?* *1.2. What are the phases in the dynamic discounting process?* *1.3. How are new invoices added to the platform for early payment?* *1.4. What are the problems with the manual early payment process?* *1.5. How are the early payment results send to the buyer?*
3. Ideal process	***Phase 2 Define objectives of a solution***	*2. What does the ideal automated process look like?* *2.1. Which process steps are to be automated?* *2.2. What are the problems in integrating the different ERP systems?*
		3. What is a suitable integration type for integrating multiple different ERP systems with a platform for early payment?
4. Canonical Data Model	***Phase 3 Design & Development***	*4. How to design and develop a CDM?*
5. Prototype	***Phase 3 Design & Development*** ***Phase 4 Demonstration***	*5. Is it possible to create a working prototype with a CDM?* *5.1. How is the prototype developed?* *5.2. What does the CDM look like?* *5.3. Does the developed prototype reduce the time needed for the data exchange?*
6. Conclusion and recommendations	***Phase 5 Evaluation***	*6. Does the introduction of an ESB with a CDM solve the project problems?* *6.1. How are the problems solved?* 6.2. *Is the solution also future proof?*
	Phase 6 Communication	*7. What can be concluded from this project?* *7.1. Is the project goal reached?* *7.2. What are the recommendations?* *7.3. What could be further researched?*

2. Current process

In this chapter, the current data exchange process between the platform for early payment and the ERP system is explained. It is important to have a clear overview of the current situation to identify what can be improved. In 2.1 the normal invoice process is described, without early payment. In 2.2 dynamic discounting is explained. In 2.3 the early payment process of Company A is displayed and described. In 2.4 the manual steps in the data exchange between the ERP system and platform for early payment are described, the time these steps take, as well as the data that is exchanged.

2.1 Invoice process

The invoice process without dynamic discounting can be divided into three phases. The invoice composition, invoice receiving and invoice payment (Perego & Salgaro, 2010). These phases are modeled in

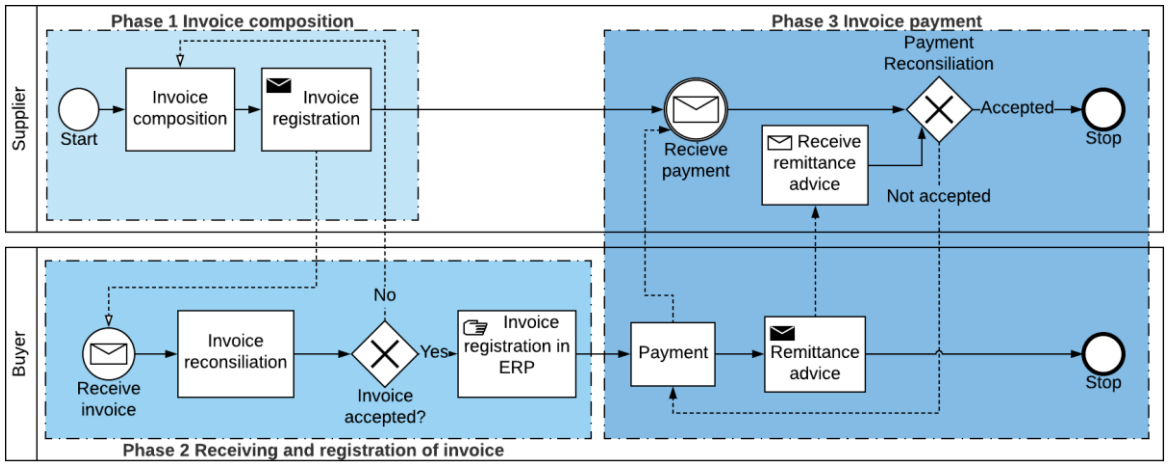


Figure 2 based on sources of Company A and CAPE Groep, using the Business Process Model and Notation (BPMN see BPMN for clarification of the used symbols).

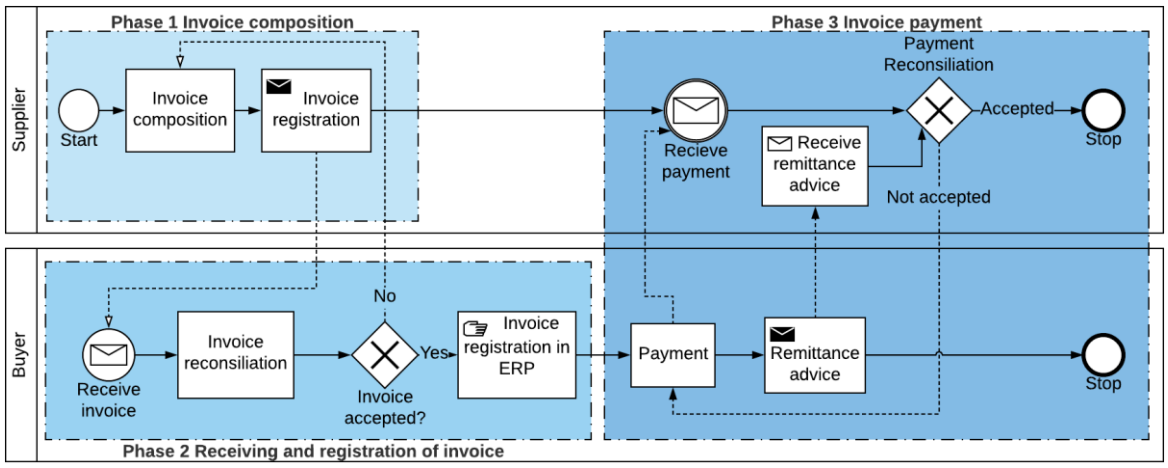


Figure 2 Invoice process without dynamic discounting

An invoice is created by a supplier after a buyer ordered a product. The invoice is sent to the buyer via the post, email, fax or Electronic Data Interchange. When the invoice is received, the buyer will check it for correctness. If there are any inaccuracies with the invoice, the supplier will get notified. Otherwise, the invoice will be entered into the ERP system and

paid at the due date in accordance with the conditions and discounts that were specified in the order.

When the payment is completed a remittance advice is sent to the supplier to notify that the invoice is paid. With the remittance advice and the invoice, the payment is verified on accurateness. When everything is correct, the invoice process is finished.

2.2 Dynamic discounting

The platform for early payment facilitates dynamic discounting between buyer and supplier. The dynamic discounting process will be explained in this paragraph, the early payment process in the next paragraph.

Dynamic discounting is the possibility to receive a discount based on the payment date of an invoice. The earlier the payment is done by the buyer, the greater the discount. Dynamic discounting can be viewed as a line with a negative slope, where the highest discount is given when the invoice is paid at the moment of receiving and no discount is received when the invoice is paid in accordance with the contractual payment terms. The discount, in percentage per day, is the slope of the line. When the discount percentage per day is constant, the line is displayed in Figure 3. The discount percentage can also decrease faster in time in order to encourage earlier payments even more (Gelsomino, Mangiaracina, Perego, & Tumino, 2016).

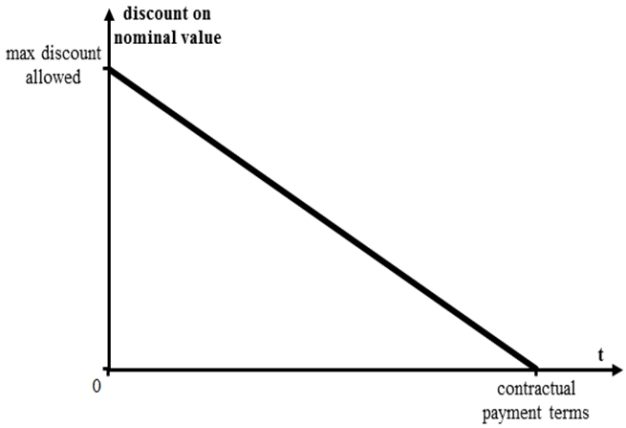


Figure 3: Dynamic discounting with a constant daily discount rate

This differs from traditional discounting is that the discount it is not a static term, but a function of the time of payment. Also, with dynamic discounting, the buyer pays directly to the supplier, as oppose to Supply Chain Finance, where a third party finances the early payment to the supplier and the buyer pays to the third party afterward. With dynamic discounting, the buyer pays the full invoice amount minus the discount to the supplier. A possible fee for using the platform for early payment is paid by the buyer.

Phases of dynamic discounting

When dynamic discounting is used, the process can be divided into four phases as defined by (Gelsomino et al., 2016).

The first phase is the invoice composition. This contains all the activities that are carried out by the supplier to create the invoice and ends when the invoice is sent to the buyer.

The second phase is the receiving and registration of the invoice. In this phase, the buyer receives the invoice from the supplier and it is checked against the delivery notes and orders. When the invoice is correct, the buyer will enter the invoice into his ERP system.

The third phase is the Early Payment Proposal issuing activity. In this phase, the buyer submits a request for early settlement of an invoice in exchange for a discount on the invoice amount. This request contains two data items: the proposed earlier payment date of the invoice and the proposed discount. The supplier can accept or decline this request. When the request is refused, the buyer can update the request to a different discount or an earlier payment date. When the request is finally accepted, or when the standard payment terms have been reached, due date and the discount will be updated in the ERP system of the buyer and at the platform for early payment.

In the fourth and final stage is the invoice payment. This phase ends when the supplier has checked that he received the correct invoice amount and accepts the payment.

2.3 The early payment process

The current early payment process of company A is displayed in Figure 4. This figure is also created with the use of BPMN and displays the whole process from the sending of the invoice by the supplier until the time the payment is received.

2.3.1 Phase 1 and 2

The first two phases of the process are the same as the process without early payment, these phases can be found in 2.2.1.

2.3.2 Phase 3 Early payment proposal

After the buyer accepts the invoice for payment, the invoice is uploaded manually to the platform for early payment. This is done in a specified format, which is described in 2.4.1 After the invoice is uploaded to the platform for early payment, the buyer has the possibility to submit an Early Payment Proposal (EPP). When the proposal is uploaded to the platform for early payment, the supplier can accept or decline the early payment. When the proposal is declined, the buyer has the possibility to update the proposal to a more appealing discount percentage or an earlier payment date. This process continues until the proposal is accepted or the payment date is reached. When the EPP is accepted, the invoice terms will be updated in the system of the buyer and the invoice is ready for payment to be archived. On the platform for early payment, all invoices with an accepted proposal are sent to the buyer at 23:00.

2.3.3 Phase 4 Invoice payment

The invoice will be paid by the buyer when a proposal is accepted, or at the due date of the original payment terms. When the payment is completed, a remittance advice will be sent to the supplier. This remittance advice describes the paid amounts and the discounts that were applicable to the invoice. When the payment is received by the supplier, it will be checked for correctness with the invoice and the remittance advice. If the payment is performed correctly, the process is completed.

2.4 Manual actions and data exchange in the current process

New invoices are uploaded by the buyer to the platform for early payment through a manual upload of a file. The steps that must be performed to upload the invoice are:

1. Select the data in the ERP system.
2. Enter the data into a file
3. Make sure the format of the file is correct
4. Log in on the platform for early payment and upload the invoice file

The time to perform these phases has been measured by timing the execution of these phases by three different people. The steps that must be performed are explained in Appendix D. The results are displayed in 2.4.1 and show that it takes a buyer approximately 5 minutes to upload an invoice to the platform for early payment.

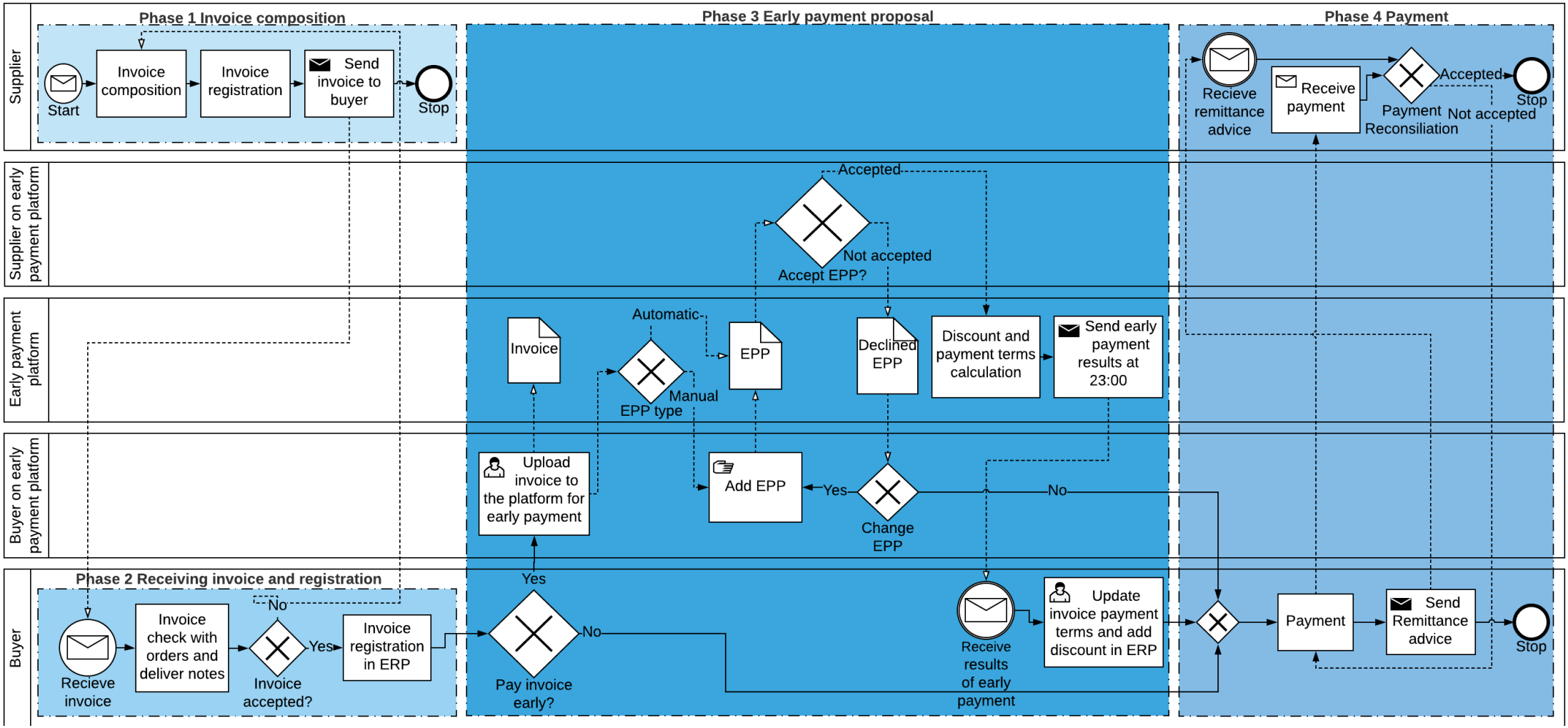


Figure 4 Current early payment process

Phases in manual upload process	Person 1	Person 2	Person 3	Average
Select the data in the ERP system.	0:10	0:09	0:15	***0:11***
Enter the data into a file	2:41	2:05	2:17	***2:21***
Make sure the format is correct	1:09	0:22	0:49	***0:46***
Log in on the platform for early payment and upload the invoice file	1:51	1:43	1:27	***1:40***
Total time	**5:51**	**4:19**	**4:48**	***4:59***

Table 1 Baseline manual invoice upload

2.4.1 Uploaded invoice data

The data that is currently uploaded to the platform for early payment is displayed in Table 2.

Fields of the invoice upload file	Format
Invoice reference number for own system	String
Supplier code for own system	String
Currency	String
Total invoice amount, excluding VAT	Decimal
Total invoice amount, including VAT	Decimal
Invoice date	DD-MM-JJ or DD/MM/JJ
Invoice reference number indicated by the supplier	String
The partial amount destined for a block account number	Decimal

Table 2 Data uploaded in the manual invoice upload

The data can be uploaded to the platform for early payment by uploading a comma separated file (CSV) or a text file with the correct formats. This is an error-prone process. Errors in typing data into a file are easily made, as well as errors in separating data fields, in file formats and in the upload process. These errors are not documented by the buyers of the platform for early payment, so there is no information available about the quality of the current upload process.

2.4.2 Data sent from the platform for early payment

When the early payment is accepted, the data (invoice number of the buyer, the agreed early payment date, and discount percentage) is sent from the platform for early payment to the buyer at 23:00, using the SSH File Transport Protocol. This is a secured protocol that ensures safe file transfer between systems.

Entering the result data into the ERP system is a difficult process and requires expertise about the ERP system because most ERP systems do not have standard fields for dynamic discounting. The payment due date can be changed, but it is not possible to just change the discount because the reconciliation of the payment depends on the correct VAT amount of the original invoice. The discount must be added via a specific after invoice discount possibility which in most ERP systems only supports discount periods. Because there are many different ERP systems, it is not possible to standardize this process. Therefore, the buyer must have the expertise on how the discount data must be added to their ERP system.

2.5 Summary and conclusion

In this chapter, the current early payment process of company A is explained. Dynamic discounting is explained, and the early payment process is divided into phases. The duration of these phases is measured as a baseline of the current process. The following questions are answered in this chapter:

1. What does the manual early payment process look like?
    1.1. What is dynamic discounting?
    1.2. What are the phases in the dynamic discounting process?
    1.3. How are new invoices added to the platform for early payment?
    1.4. What are the problems with the manual early payment process?
    1.5. How are the early payment results send to the buyer?

Dynamic discounting is the ability to receive a discount for every day an invoice is paid before the original invoice due date. This makes it possible for the buyer to use excess cash and for the supplier to improve his working capital. Dynamic discounting is used in the early payment process of company A, which consists of four phases: Invoice composition, invoice receiving, early payment proposal and invoice payment. Within these phases some tasks are currently performed manually by the buyer: the uploading of invoices to the platform for early payment and the processing of the results from the early payment process into the ERP system. In the current process, it takes on average 5 minutes to upload an invoice to the platform for early payment and the process is vulnerable to errors. The results from the early payment process must be entered into specific fields in the ERP which is difficult and different for every ERP system.

3. Ideal process

As described in chapter 2, the invoice data is currently added manually to the platform for early payment. Also, the results from the platform for early payment are sent from the platform to the system of the buyer manually.
In 3.1 the automation of this data exchange is described. It will be explained that automation of this data exchanges requires data integration. In 3.2 various types of data integration and their characteristics are described, and the middleware integration is explained for the connection of the ERP systems with the platform for early payment.

3.1 Automation of data exchange

To eliminate the manual steps in the exchange of the data between the platform for early payment and the different ERP systems, this exchange must be automated. The automation of the upload of the invoices will reduce the time that is needed to start the early payment process. After this process is completed, automating the entering of early payment results into the ERP system of the buyer will simplify the updating of the payment terms and save the buyer time.
The steps that must be automated are displayed in orange in Figure 5. To automate these business processes, the systems need to be integrated to support interactive automatic communication between these systems. A problem with this integration is that all ERP systems of the buyers have a unique structure of the data and the naming and types of the data fields are different in every system. Moreover, the names of the fields and tables are multi-interpretable abbreviations and vary per system. Therefore, integrating ERP systems with the early payment system is a delicate task that needs expertise from both systems, comes with high costs and has low flexibility. Also, the integration with one ERP system can only partly be reused for the integration of another ERP system (Lemcke, Stuhec, & Dietrich, 2012).

3.2 Data integration

In this paragraph, the different types of integration will be explained. Based on these types, the integration type for integrating the platform for early payment with the ERP system will be chosen.

Before the different types of integrations are described, we will first define what an integration is. An integration is an interactive process of combining subsystems to form one system that is consists of several systems that function as one. System-To-System integration is the flow of data from one system to another system (Gulledge, 2006).

There are different ways to integrate the ERP system and the platform for early payment. To integrate systems Land & Crnkovic (2003) propose four different types of integration:

- Interoperability by **data import and export** facilities. This approach can be done manually or automatically and allows data to flow between the systems. The problem with this approach is possible data inconsistency as files could get uploaded twice or forgotten and the format could be different in de import and export system.

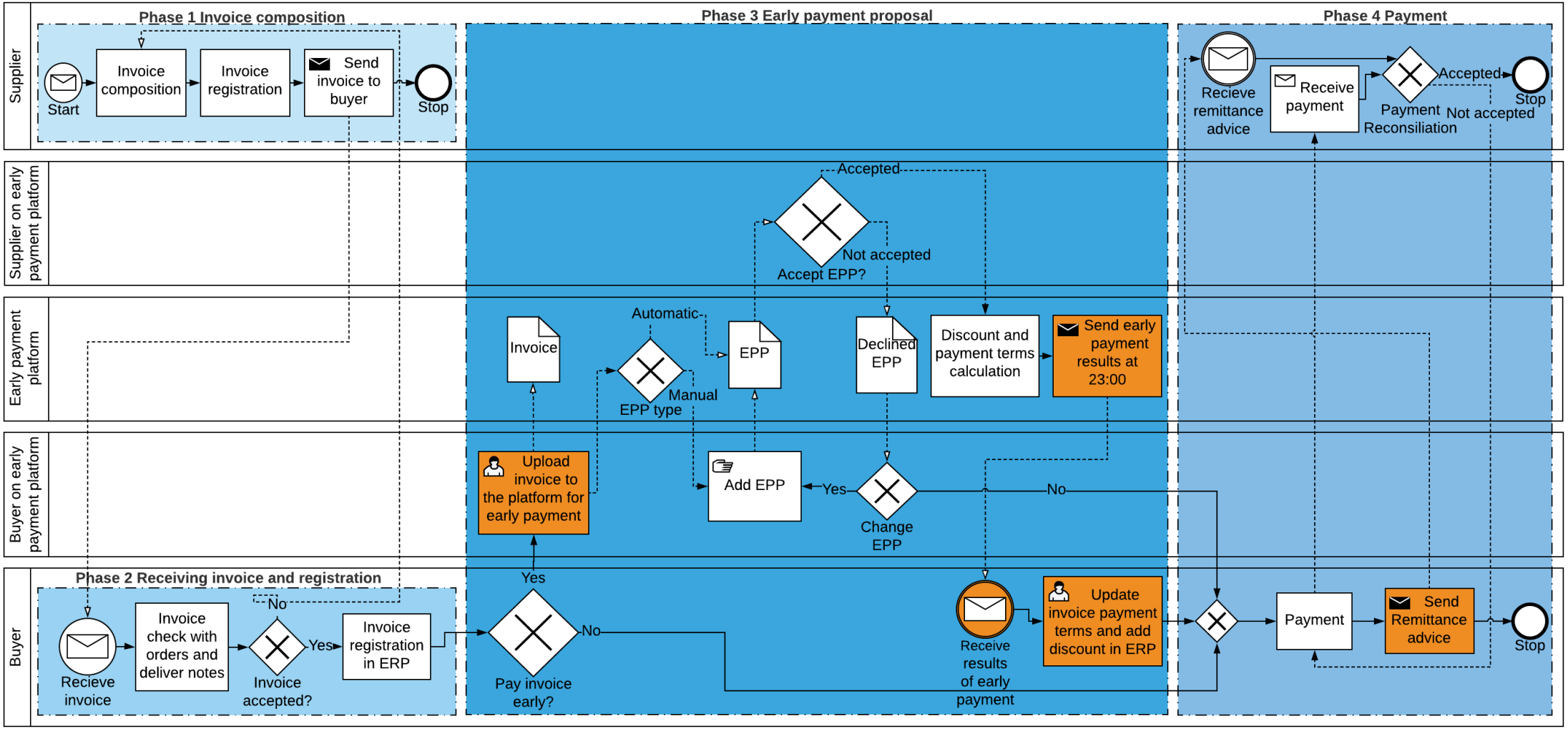


Figure 5 Early payment process tasks for automation

- With **data-level integration**, data will be shared through a common database. This makes data accessible for all related systems. The data is consistent, coherent and correct. To implement this, a common database model must be modeled, and the existing systems must be modified to use this common database. The maintenance of related systems will become complex because each data-level change has an impact on all other coupled systems.
- The third option is **Enterprise Application Integration** (EAI). In many company's systems are bought and the source code or databases cannot be changed. To connect these software systems a middleware is introduced. Systems can connect to this middleware via custom wrappers, adapters or other connectors that must be built for every system. This enables a so-called "loose" integration, where systems don't need to know the details of how to send data to another coupled system. Systems can then store their data independently and in their own repository (Lee, Siau, & Hong, 2003).
- Systems can also be integrated at **source code level**. Different systems will behave at the user level as if it were just one system. With bought ERP systems, this is often not possible, because the source code cannot be changed.

Currently, the systems are integrated with the first integration type, import & export of data, because the invoices are imported via a .txt or .csv file and the results from the early payment are exported into a file, which is sent to the buyer.

To automate the upload of invoices and to process the results of the early payment process a common database could be used. This second integration type requires ERP systems and the platform for early payment to be connected to this database. But this requires reprogramming of the ERP systems and the platform for early payment to use this database for the storage and retrieval of the data. Also, all coupled systems become dependent: a change on data-level in one system affects all related systems. This option is not applicable in this situation.

Because the platform for early payment is designed to be used with multiple ERP systems, it is inconvenient to implement the fourth integration type as well. This type of integration will require the functionality of the platform for early payment to be programmed into every ERP systems. This is very difficult due to the different source codes of the systems and must be done in every ERP system.

The third integration type, Enterprise Application Integration, lets the ERP systems and platform for early payment store their data independently. The integration of these systems is accomplished by creating adapters for the ERP system and the platform to connect with an interapplication middleware (Lee et al., 2003). This type makes it possible to connect multiple different systems and with future systems that are not yet known. This is not possible in integration types two and four. This makes it the best type to integrate the ERP systems with the platform for early payment.

3.2.1 Introduction to Enterprise Application Integration

When two systems are integrated using a direct link, this is called a point-to-point integration. When a point-to-point integration is used, and all the systems are integrated, the number of integrations is N x (N - 1) for N systems, see Figure 6. Point-to-point integrations are vulnerable to change. This is because these integrations are programmed into the systems and every change requires reprogramming of these interfaces and their underlying logic (Weske, 2012).

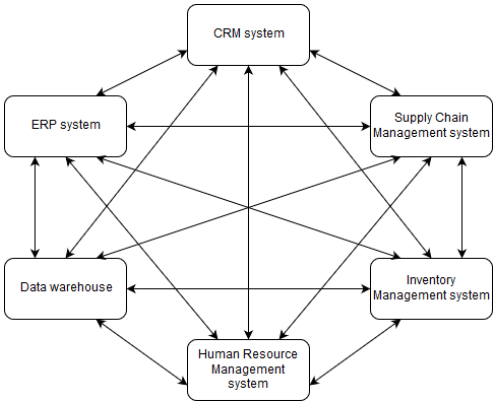


Figure 6 N x (N -1) integrations

With the creation of a centralized middleware as in Enterprise Application Integration (EAI), the point-to-point integrations are replaced by connections. A connection is a link from a system to a central middleware. This way the number of connections can be reduced to N connections for N systems.

An Enterprise Service Bus (ESB) is a type of middleware that combines principals of messaging, transformation, translation and web services. CAPE Groep uses an ESB eMagiz (see 5.3.2). The ESB is an integrations platform using common standards, that connects and manages the integration of systems with different data structure and representation. Systems are connected with adapters, these adapters must be developed specifically for every system. Within the bus, the messages are routed to the correct receivers based on the content of the message (Chappell, 2004).

The ESB can solve the difference between data structure and representation of the systems with the use of a data model in the bus that all applications agree with. Using this common model data can be mapped to the data structure of the application. This common data model is called a **Canonical Data Model** (Teusch, 2014).

With the ESB, the number of connections between the input and output systems can be reduced from X * Y (where X stands for the number of input systems and Y for the number of output systems) to X + Y. With the ESB, when one Y system changes only its connection to the EBS must be updated instead of X connections from Y to X in case of point-to-point integration (Hohpe & Woolf, 2004).

The ERP systems are used by different buyers, which means that they all store their data independently. To connect with the bus, they require custom adapters to communicate with the bus. Translators must be added to these adapters to convert the data into the canonical representation that is used in the bus (Chen, 2012).

3.2.2 N to 1 integrations

When there are N systems that must be integrated into one system, the number of the system integrations does not change when an EAI integration type is used. This number will still be N*1. The number of connections, however, changes from N to N+1 for the EAI solution, as all N systems must connect to the middleware and from the middleware, one connection must be made to the one system, see Figure 7. This extra connection, however,

is simple to create because the connection must be made from the bus to the system. To create this connection, the creator only must have knowledge about the system that must be connected to the bus, because the bus uses the CDM which is a representation of all systems and should be easy to understand for all systems. The EAI type is also useful for when Company A would extend their services and an extra system is added on the right. Then the number of extra connections with EAI is only one and with point-to-point this would be equal to the number of ERP systems on the left.

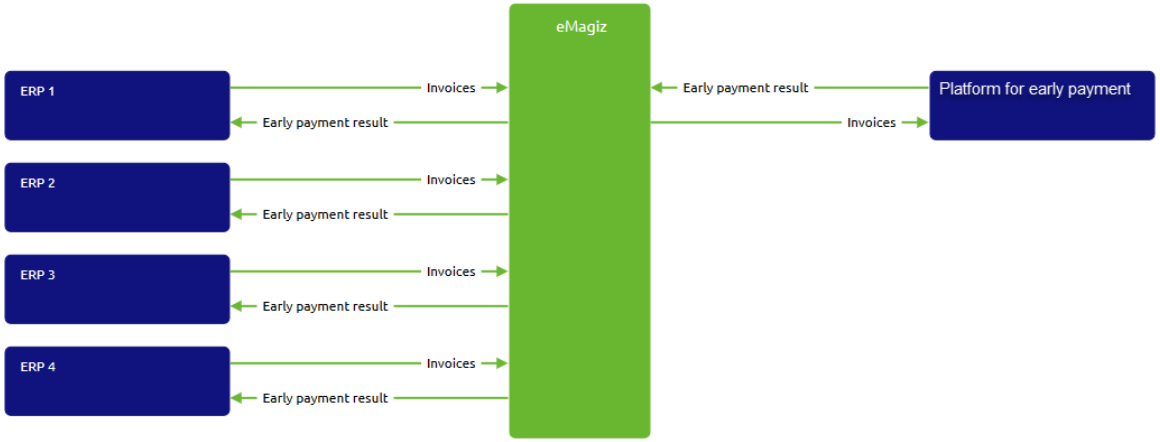


Figure 7 Number of integrations and connections

When a system sends a message to the bus, the message is translated to the CDM and then translated from the CDM to the correct format of the designated system. The method that is used to transport the message to the bus has no influence on the message that the destination receives because the message is translated to the CDM which only requires the message to have the correct data and does not care about the names of the data.
In the next chapter, the CDM will be explained and how a CDM can be designed and developed will be discussed.

3.2.3 When to use an Enterprise Service Bus

1. **Two systems**
   When one system must be integrated with another system, the use of an ESB is **not** practical. A CDM would then have to be developed and the connectors to this CDM. These systems could better be integrated with a point-to-point integration as the transformation is then made only once and must be understandable for only one system.
2. **Two systems with one system**
   When two systems are integrated into one system, the use of an ESB is often **not** useful because the number of integrations is lower with point-to-point is (2) than with an ESB (3). When the one system that the two systems are integrated with is changed frequently, with every change the two integrations must be changed. With an ESB only the integration to the bus would have to be changed.
3. **Three systems to each other**
   When three systems must be integrated with each other, the number of integrations is the same for point-to-point and ESB. An **ESB is in this situation better** than point-

to-point because when one system changes only one integration must be updated where two integrations must be updated with point-to-point. An ESB is also better when in the future an extra system might be added.

4. **Three systems with one system**
   When three systems must be integrated with one system, the use of an ESB requires one more integration compared to point-to-point but the **ESB is better** for when a system changes and when a new extra system must be integrated with the one system.
5. **More than three systems with each other**
   An **ESB is a good method** for integration when more than three systems must be integrated. The creation of a CDM is better than creating individual integrations. The CDM also supports flexibility and the ESB makes the integrations easier to change in the future. Figure 8 shows the number of integrations of point-to-point or ESB per number of systems.

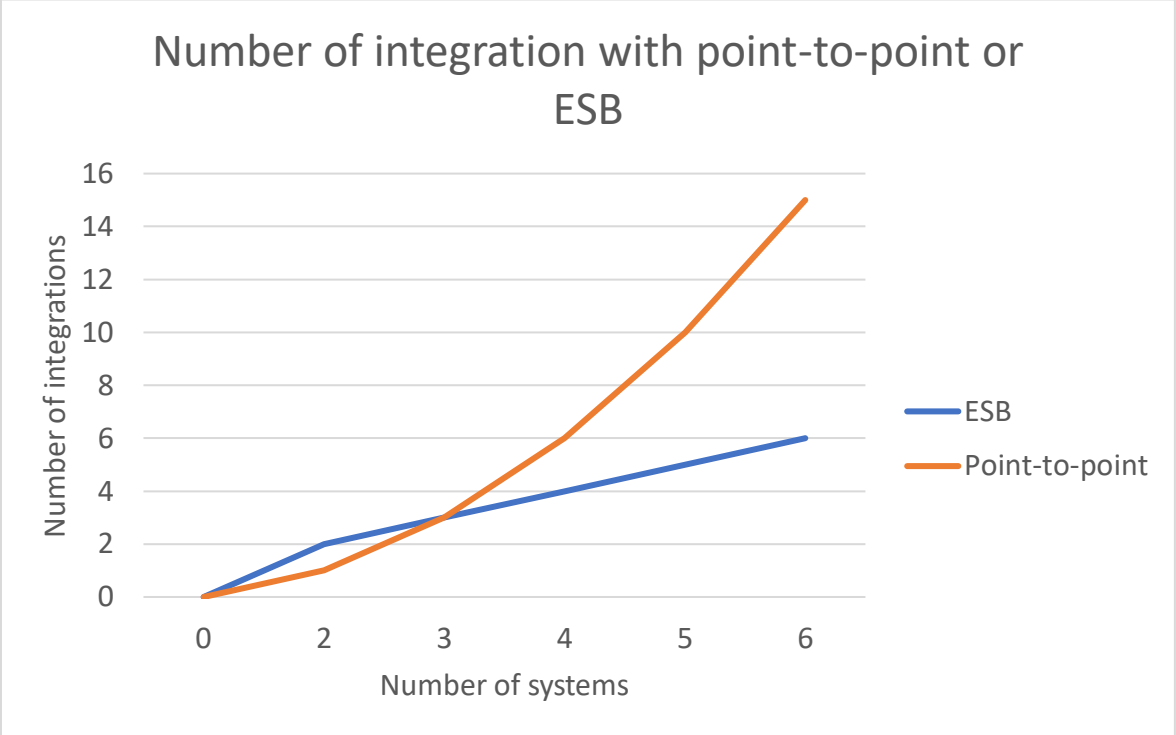


Figure 8 Number of integration with point-to-point or ESB

When there is one system that must be integrated with N systems the use of an EBS functional especially when the system that must be integrated with all the other systems is still changing, due to development for example. When the system changes only the adapter and the mapping to the CDM would have to be changed for one system and not for every connection as when the systems would be point-to-point integrated.

A problem with data integration is that, when the fields in a system are not unambiguously filled, errors will occur. With an ESB these errors are collected, analysed and solved at the bus.

3.3 Summary and conclusion

In this chapter, the ideal process of the data exchange between the ERP systems of the buyer and the platform for early payment is described. The need to automate the manual steps that are currently performed is explained. To achieve this, different integration types are discussed to select an integrated type for integrating the different ERP systems of the buyers to the early payment process. The questions that are answered in this chapter are:

2. *What does the ideal automated process look like?*
   2.1. *Which process steps are to be automated?*
   2.2. *What are the problems in integrating the different ERP systems?*

3. *What is a suitable integration type for integrating multiple different ERP systems with a platform for early payment?*

The ideal automated process is that the manual steps are automated. The steps that are automated in the ideal situation are displayed in orange in *Figure 5*. The problems with integrating the systems are that every system uses a different data structure and have different naming of the fields and tables. The current integration type is the import and export of files. The EAI integration type is identified as a better alternative that can be used with multiple current and future systems. The use of the Enterprise Service Bus is found to be an Enterprise Application Integration method that facilitates the integration of systems with different structures and different data representations. When three or more systems are to be integrated, the number of integrations that must be created is less with an ESB than with point-to-point. Next chapter will discuss the Canonical Data Model (CDM) of such an Enterprise Service Bus.

4. Canonical Data Model

In the previous chapter, the Enterprise Application Integration type with an Enterprise Service Bus is introduced as a method to integrate multiple systems. In this chapter the central data model of an ESB, also referred to as the Canonical Data Model (CDM), will be described, as well as how this model can be created and the advantages and disadvantages of this model. In paragraph 4.1 the topic of standards will be introduced. In 4.2 data heterogeneity will be explained. In 4.3 the design and development of a CDM is described and in 4.4 the advantages and disadvantages of a CDM are defined.

4.1 Literature review

To answer the fourth research question:

4. How to design and develop a CDM?

A systematic literature review is performed to find literature relevant to this topic. The literature is searched in the Scopus and Google Scholar databases. The search terms that are used are: “Canonical Data Model” AND “design”, “Common Data Model” AND “design”, “Canonical Data Model” AND “development” and “Common Data Model” AND “development”. Both the Canonical Data Model and Common Data Model are used because they are synonyms (Raap, Iacob, van Sinderen, & Piest, 2016). The review protocol, inclusion and exclusion criteria and results can be found in Appendix A.

4.2 Standards

To simplify system integrations there are a lot of different standards and protocols for every domain. These standards and protocols are designed to create flow and allow businesses in a certain domain to communicate without the need of translation (Roman, 2006). Because every organization uses a different selection of these standards and protocols, templates are created for companies to ease the use of standards. This results in large templates that contain thousands of fields to communicate within a certain domain. These templates are very big because they focus on completeness to cover all requirements. From these templates, only a small part of the fields are used frequently (Dietrich, Weissmann, Rech, & Stuhec, 2010). Because there is little knowledge about the data similarity in different domains, making use of standards only helps partially with connecting businesses and doesn’t solve the problem of connecting and integrating with different industries that use different standards (Dietrich & Lemcke, 2011).

4.3 Data heterogeneity

Because most standards use their own data structure, with names and types, there are many different representations of the data. These differences of representation in names and types are called data heterogeneity. The semantic differences, that occur due to data heterogeneity, need to be sorted out when systems are integrated. This can be done by creating a common understanding (Weske, 2012).
To create this common understanding, a CDM can be developed. A CDM is created to capture the semantic equivalence between all schemas. This equivalence is captured in the canonical model just as the deviation from the equivalence. The CDM can be understood as

the bridge between all schemas, which makes it easier for the integrators of different systems to understand the CDM (Dietrich & Lemcke, 2011).
By creating a single view of all the independent systems, this view can be used with little adaption to connect to any other system. It takes an iterative process to build this model independent from the order in which the schemas are imported because the first schemas that are added have a bigger impact on the model, which may result in less correspondence with the schemas that are added later (Lemcke et al., 2012). To create this independent abstract overview model is the hardest part of creating a CDM.

4.4 Creation of a CDM

In the industry and academia, some attempts have been performed to solve the integration of different standards. These attempts of the industry propose solutions that are based on XML and XML schemas. But these attempts only focus on the extension of a standard and not on the transformation between standards. Besides manual mapping, no mechanism for the support of the mapping between the schemas is presented. Therefore, the problem of integrating the schemas remains with this approach (Roman, 2006).
In academia, (semi) automatic approaches for the integration are proposed. These approaches are schema-based and ontology-based. The schema-based approach uses schema matching which is defined by Michael Dietrich & Lemcke (2011) as "the semi-automatic finding of semantic correspondences between elements of two schemas".
Schema matching is performed automatically by starting at the root attribute of the schema and compare each attribute between the target and the source schema. This procedure results in a table which can provide information about which attributes in the source schema must be mapped to which attribute in the target schema (Dietrich & Lemcke, 2011).
With this schema-based automated approach, the common model can be iteratively improved and adapted when new schemas are imported. The knowledge from the mapping between incorporated schemas can be reused since previous connections can predict the mapping with a new schema. This mapping can, therefore, be derived from the table of correspondences (Dietrich, Lemcke, & Stuhec, 2013).

The ontology-based approach develops a high-level semantic understanding of the schemas to create a model that covers all the semantics of the underlying schemas.
Ontology-based is identified as superior to schema-based, but there is no clear approach on how to use the ontology in different schemas to find a mapping between the different standards. The advantage of the ontology-based approach is that it provides a flexible way to create a connection between the informal world of business standards and the formal data model. Ontology-based integration eliminates the need for schema matching because it can be used on a higher level and is schema independent. To create an ontology-based matching, therefore, requires a discovery of the semantic correspondence which can be created with language interpretation research (Roman, 2006).

Roman (2006) states that during the creation of a CDM humans play a key role in integrating because they need to understand the sometimes-ambiguous specifications and meaning of the different standards and make them work together. The human builds the common understanding of the standards in the model. This enables the interoperability between

these different systems through the model. The problem is that the common model is created in the human mind and is only represented implicitly. Because of this, the model is created in an ad-hoc way, with no structured method. This makes it difficult to create the model faster and achieve a less expensive creation of the model as also no structured method is proposed by the industry or academia (Roman, 2006).

When the models could be created explicitly, the common model could be designed with more structured and clear methods. To achieve this further research on structured methods for humans on the creating of a Canonical Data Model would have to be done. A structured method for support of the construction of a Canonical Data Model is useful because then the model would be based less on the intuition of the creator of the model.

When creating a CDM, a language is needed in the model to flexibly manage the different standards that are represented in the model. This language should be generic to make the integration between the different standards possible and should support the transformation between the different standards (Roman, 2006).
In the CDM the names of the fields of the CDM should be readable and consistent. The names of the attribute labels can be identified from the original name of the attributes or the naming could follow a common standard. Consistency and expressiveness of the names is useful for when schemas use cryptic labels (Dietrich & Lemcke, 2011).
When integrating schemas from different domains and languages, there are different type definitions. With the creation of the CDM, one has to be aware of the differences when transforming from the source schemas to the CDM, because the source schemas may use different type definitions (Smith & McBrien, 2006).

4.5 Advantages and disadvantages of a CDM

The platform for early payment is developing over time. This means that changes will occur, and these changes can affect the common data model. Because the common data model should preserve the semantic correspondence between all the data the ERP systems and the platform for early payment send to the bus, these changes to the CDM will likely be small or the data is already included in the CDM.

The CDM creates a higher level of abstraction, because of the common model, the integrations are not entirely based on the field names in the individual systems. The mapping is now based on which data needs to be integrated. This requires a semantic understanding in the model of the data that is sent to each system. When this semantic understanding is created in the model, it eliminates the need to talk about abbreviated field names of the data that do not describe the full contents of the fields. The CDM resolves the semantic dissonance as it is only interested in the context of the fields and does not focus on the field names in the individual systems (Hohpe & Woolf, 2004).

With the CDM only one connection must be made from each system to the model. This means that the integrator only must understand field names in the system that has to be connected and the field names in the common data model. The common data model is generic and should, therefore, be easily understandable.
The integrator then does not need to have any technical understanding of the data model of

the system that it has to be integrated with. This resolves a lot of problems and makes maintenance easier, as an update will only involve his own system and in some cases the common data model.

When the data connections to the bus are described in a CDM, this creates a better overview of which data is available for use in the other system. This makes it possible for company A to see which data is available from the ERP systems and what impact an update of the platform for early payment will have on the communication with these ERP systems.

The CDM must work equally well for all the systems that it communicates with. But to achieve this, the CDM doesn't have to be the model of the complete database of each coupled system, but only of the portion that is involved within the messaging.

A disadvantage of the CDM is that the data must be translated to the CDM and from the CDM. This creates an extra translation compared with a point-to-point integration which adds extra latency. This makes the use of an Enterprise Service Bus often slower than a point-to-point integration (Hohpe & Woolf, 2004).

The creation of a CDM is often difficult because the semantics of the data in the schemas that are to be integrated is often only completely understood by the designers of the schema. These semantics are not clear from the schema itself (Madhavan, Bernstein, Doan, & Halevy, 2005).

Because there is no structured method for the creation of a CDM, the process of creation is often long and requires research about the understanding of the schema and intuition for the selection of the fields that must be included in the CDM. This results in a CDM that is often the interpretation of the different schemas of the CDM creator, this interpretation can differ per creator.

4.6 Summary and conclusion

In this chapter, the creation of a CDM for integration between different standards is described. How a CDM should be created is described in this chapter with the following research question:

4. How to design and develop a CDM?

For the creation of a CDM there is no structured method identified in the literature. However, to simplify system integrations, many different standards have been created. There is little knowledge about the similarity between these standards which makes it difficult to integrate between different standards. Standards use their own data structure and semantic understanding which needs to be unified when integrating. This can be done by creating a common understanding in a CDM. In the industry, standards are proposed to be integrated with XML, but only extensions of a standard are proposed and no transformations between these standards. In academia two (semi) automatic approaches for integrations between the different standards are proposed. The first schema-based focusses on the finding of correspondences between the elements of two standards by comparing their schema. The second ontology-based approach develops a semantic understanding, it

provides flexibility but no clear method for development is defined. Humans are essential in CDM creation as the common understanding is created in the human mind implicitly. There is no structured method which makes this a difficult and expensive process. For the creation of a CDM the language used in the model should be generic to create flexibility, the names of the fields in the CDM should be consistent and the data types must be checked for differences between schemas. The advantage of the CDM is that it is easy to understand for all the systems that must be connected to the model because it is generic. The disadvantage is that because there is no structured method for the creation of a CDM, the common understanding is difficult to create.

5. Prototype

In this chapter, the creation of a prototype for the integration of the platform for early payment of Company A with one ERP system is described.
In 5.1 the goal of this prototype is described. The requirements that must be met to achieve the goal are defined in 5.2. In 5.3 the methods that are used for the development of the prototype are described and the prototype itself is described in 5.4. The tests of the prototype are described in 5.5.

5.1 Goal

The goal is to create a prototype to prove that the integration of data between the ERP system and the early payment can be automated using an Enterprise Service Bus. The prototype will include a CDM that will be the common data model for the integration of all the systems. Because the ERP systems do not have specified fields for the results of the early payment process, a specific adapter must be programmed to integrate this data. This must be done by an expert of the ERP system. Therefore, the prototype will only focus on the full automation of the invoice upload.

5.2 Prototype requirements

Because the time for development is limited, a Minimal Viable Product (MVP) will be created, this is a product that aims to receive the maximum amount of feedback with the least effort (Taibi & Lenarduzzi, 2016). The requirements for this MVP are:

- Invoice data is retrieved from the Exact ERP system and entered in the platform for early payment without any user involvement.
- The CDM is based on the GS1 standards for invoices.
- The prototype will use version 1 of the API of the platform for early payment that was published on the 7th of June 2018.
- Besides the GS1 standard, the data structure of the ERP systems Exact and Navision must be used for the creation of the CDM.
- The improvement of the prototype is measured by comparing the user tests of the manual upload of invoices in 2.4.1 with tests of the prototype.

5.3 Development

For the prototype, several development tools and methods are used. These will be explained in this paragraph.

5.3.1 Scrum

The Scrum framework is a method for a team to work autonomously by specifying several roles. The framework enables the team to have an overview of the progress. This makes it possible for the team to redirect the focus during the project to keep working towards the goals of the solution.
In Scrum, people work in teams. Teams consist of 5 to 9 people with different roles. Teams work in sprints, which are periods of 2 to 4 weeks where at the end of the sprint a working product must be delivered. The advantage of this approach is that there is always a working prototype that can be extended with new functionality.

An important role in Scrum is the product owner. He is the owner of the problem that must be solved, for him or other stakeholders. The product owner and stakeholders often do not exactly know what the solution to the problem should be, but they are able to define their objectives of the solution. The objectives of the solution are described as user stories. These user stories together form the backlog where all the functionality that the solution should provide is documented (Sutherland, Solingen, & Rustenburg, 2011).
A user story describes a piece of functionality that must be included in the prototype. User stories are written from the perspective of the end user of the product. These stories follow a standard format: As a <*type user*>, I can <*function*> because <*added value*>
The product owner creates from the backlog a sprint backlog, this defines which user stories are to be completed in the sprint. It is important that the sprint backlog is not too big, to enable all user stories to be finished at the end of the sprint.
After the sprint, a demonstration of the working product is given. Based on this demonstration new user stories can be created for the next sprint.

For the creation of the prototype in this project, the framework of Scrum is partially followed. The product owner was represented by the supervisor of CAPE Groep, the user stories were created to reflect the tasks in the different stages in eMagiz (explained in 5.3.2). The backlog of the user stories can be found in the Appendix E.

5.3.2 eMagiz

The prototype is created in eMagiz. This an iPaaS (Integration Platform as a Service) with a graphical user interface is created by the CAPE Groep to integrate systems. eMagiz follows the principles of an Enterprise Service Bus. In eMagiz integrations can be developed in the cloud, which eliminates the need to buy hardware and software. Integrations are created with the Integration Lifecycle Management. This approach consists of five phases:

- **Capture:** In this phase, the requirements are entered: what the integration must be able to achieve. The number of messages, the spread, how they are transported, their content and security are specified to be used later in the project.
- **Design:** Here the CDM is created and the messages of the systems are mapped to the CDM. These mappings form the base for the data transformation that must be done in the messages.
- **Create:** The mappings are optimized, and the routing of the messages is added. The messages are validated against the structure of the incoming and outgoing messages.
- **Deploy:** When the mapping and routing are created, the flows can be deployed to start the flow of the messages.
- **Manage:** When the transformations are deployed, they can be managed and monitored in the last phase. This helps to identify problems and to observe the usage of system resources.

5.3.3 CDM creation

For the creation of the prototype, the EAI integration type is used with an ESB and a CDM that has a correspondence with most ERP systems. This is because the ESB supports multiple methods of transport to the bus and the CDM creates a common understanding of the data that is sent to and from the bus. The database structure of most ERP systems is similar, but

there are great differences at field level. No structured method for the creation of a CDM was identified in 4.4 Creation of a CDM. Therefore, the CDM was created intuitively by comparing a specified generic language with the schemas of the ERP systems and the API. The generic language that is used for the integration between the different standard and schemas in the prototype is the GS1 standard for invoices. This standard is chosen because it is industry independent and it provides an independent common understanding of the fields that should be included in an invoice and it helps with the creation of the names for the fields in the CDM. To create the CDM, the schemas of GS1 XML where compared with the data schemas of Exact and Navision, which are 2 of the most popular ERP systems in the world. By comparing the schemas for correspondences and selecting the fields that existed in all the schemas the CDM was intuitively created. After this comparison, the CDM was updated by adding the fields that are used in the API of the platform for early payment and the fields that were not required were made obsolete.
For the CDM to be a complete representation of the ERP systems and the platform for early payment, the data about the payment methods, payment terms, vendors, remittance advice, invoices and the early payment process results are represented in the model.

5.3.4 GS1

As specified in the requirements, the CDM will be based on the GS1 standard. This standard is provided by GS1.org. This organization provides a list of well-defined concepts for customers, suppliers, and partners to communicate with each other. When each party communicates using its own processes and systems, it can be hard to get a common understanding. The communication will be time-consuming and error-prone. By providing standards for communication within a supply chain, businesses can save time and money when the standards are implemented. The use of a standard improves efficiency, safety, and visibility for supply chains on their physical and digital channels. A well-known standard in GS1 is the barcode. But GS1 also focusses on standardization of electronic messages. The GS1 standards are developed an maintained by a community of representatives from different industries (GS1a, 2012).
In the creation of the CDM, the XML standards of GS1 for an invoice will be used as a basis for the data that must be included in an invoice.

5.4 Functionality prototype

In this paragraph, the prototype that is created will be presented and explained.

The CDM that is based on the GS1 standard and the Exact and Navision ERP systems is displayed in Figure 9 Canonical Data Model prototype.

The entities and attributes that describe the invoice are displayed in Appendix E Figure 19 and Figure 19. The attributes that are not italic are required in the platform for early payment. The attributes that are italic are added for the systems to have the possibility to map to them. This makes it possible for the early payment to use them in the future and to have an overview of all the data that is available for future updates of the platform for early payment.

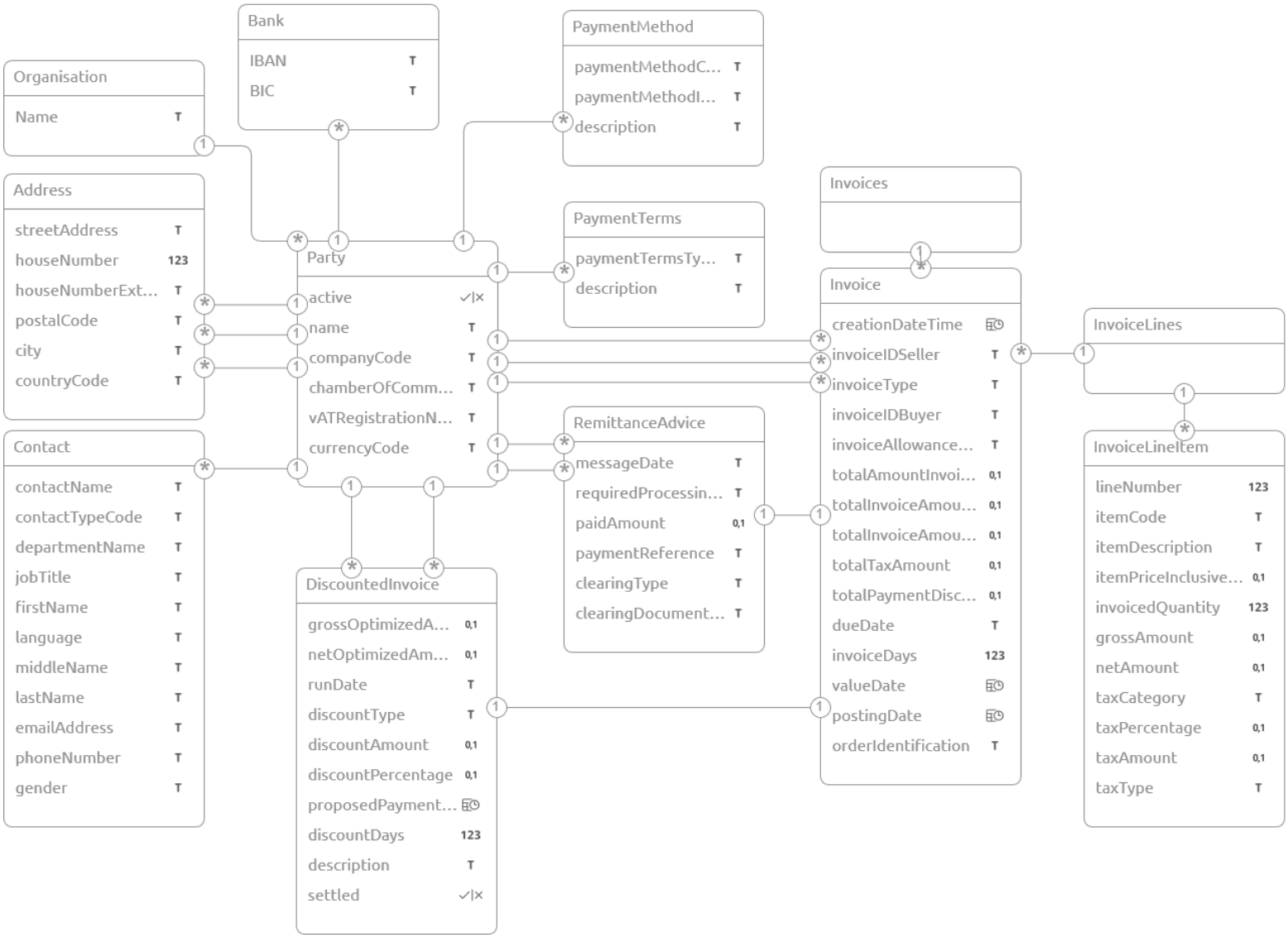


Figure 9 Canonical Data Model prototype

To the retrieve, the invoice from Exact a web service is used. This web service pulls the specified data from the database of Exact. The fields that are extracted with this specified request are defined in the Web Service Description Language. The message that is received is displayed in Appendix E Figure 21 and Figure 22. With the web service, all the invoices with a modification date of within 1 day from the retrieval are fetched.

After the message is received from Exact with the use of the web service, the message must be mapped and transformed to the CDM. The mapping is displayed in Figure 23 and Figure 23 in Appendix E.

When the message is sent to the CDM, the message must be mapped and transformed to a message to the platform for early payment. See Figure 24 in Appendix E.

After the message is transformed to the correct format, the message is stored in an XML file in a specified folder on the computer. The final message created is displayed in Figure 25 in Appendix E. The format is the same as the API. The API uses a 512-bit encryption for authentication, which is under development by eMagiz at the time the prototype was finished. Therefore, the prototype will not be able to upload the message to the platform for early payment.

The architecture of the prototype

The architecture of the prototype is displayed in Figure 10. Explanation of the ArchiMate language can be found in Appendix B BPMN and ArchiMate.

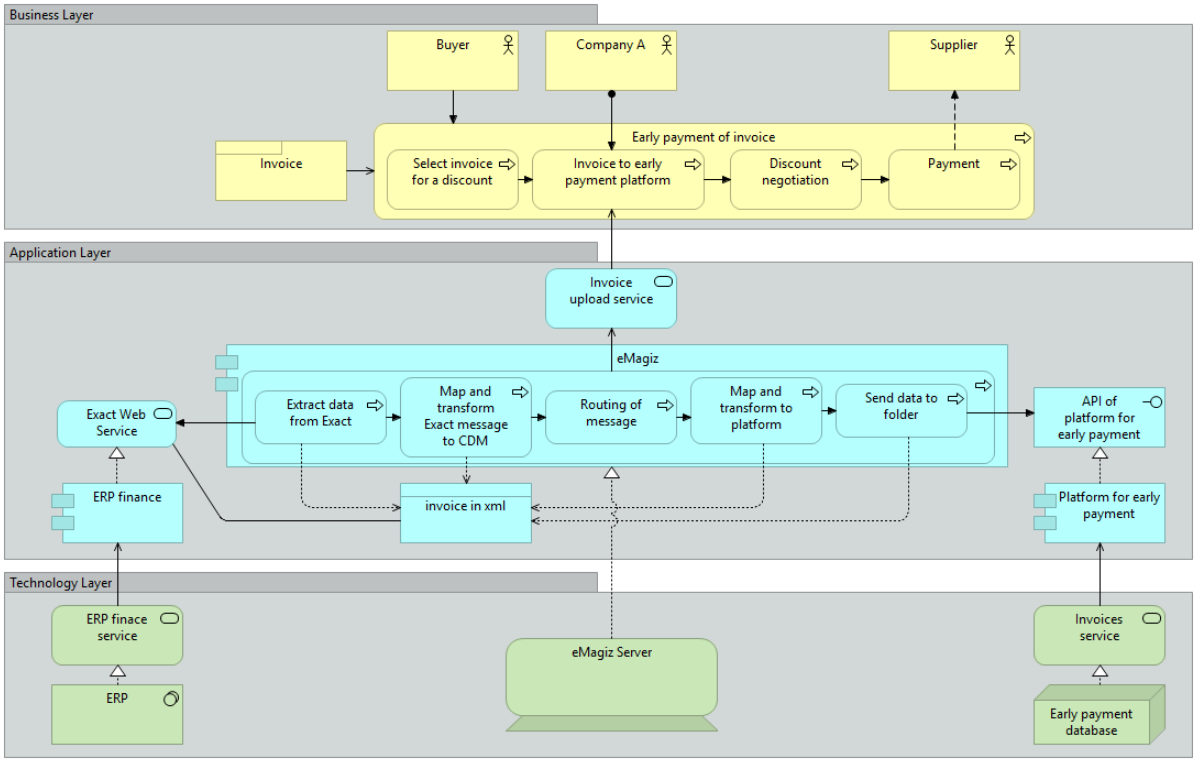


Figure 10 ArchiMate architecture

5.5 Prototype validation

The prototype performs the whole process of retrieving the invoice data from Exact until the storage of the XML file on the computer without any user involvement within 1 second. If multiple invoices are retrieved, they are all stored in the correct format in an XML file. The number of invoices that is retrieved doesn't influence the total time that is needed to perform the automated invoice process.

The duration of the manual invoice upload process was on average 3 minutes and 19 seconds without phase 4 the uploading of the invoice to the platform for early payment. The manual steps that had to be performed in the upload process are automated with the prototype. This means that the goal of upload the invoices from the ERP to the platform for early payment without user involvement is reached. Besides no user involvement, the prototype performs this the upload process in about 1 second. This is a big difference

compared to the manual upload process and proves that the prototype has a significant effect on the total time of the process.

Also, the quality of the process is improved as the possibility of typing mistakes is eliminated because the data retrieved is the same as the data in the database of the ERP. Transfer mistakes are eliminated because when the connection is set up correctly once, the transfer of data is done by the bus. Format differences are covered by the transformations described in the CDM and are therefore solved. With the automated process also procedure mistakes are eliminated, as the steps are always executed automatically in the right order.
With the manual process, invoices could be missed. In the prototype all the invoices that are changed in the last 24 hours are retrieved every hour, this makes it impossible that an invoice is missed. Duplicates are checked for changes on the platform for early payment when the invoices are uploaded and possibly updated.

5.6 Summary and conclusion

In this chapter, the prototype that is created for this project is described. The questions that are answered in this chapter are:

5. Is it possible to create a working prototype with a CDM?

- *5.1 How is the prototype developed?*
- *5.2 What does the CDM look like?*
- *5.3 Does the developed prototype reduce the time needed for the data exchange?*

The goal of the prototype is to prove that the data exchange between the ERP systems and the platform for early payment can be automated with the use of an Enterprise Service Bus. The requirements of the prototype are that it must perform the invoice upload process automatically without any user involvement, the CDM must be based on the GS1 standard and the data that must be sent to the platform for early payment must be based on the API version 1 of company A. The prototype is developed with Scrum and eMagiz. Scrum is a framework that helps with agile development and eMagiz is the ESB that is used by CAPE Groep to integrate systems. Because no structured method for development of the CDM was identified, the CDM is created intuitively by comparing the GS1 standard, that was used as a generic language, and the schemas of the ERP systems. The elements that existed in all schemas were added to the CDM.
The prototype performs the invoice process in about 1 second which is a big improvement over the manual upload process. Also, the quality of the upload process is improved as mistakes in formats, forgotten invoices and procedural differences are no longer possible.

6. Conclusion and recommendations

This final chapter evaluates the project goal and research question from paragraph 1.3 and 1.5. The conclusions of the project and research are described in 6.1. Based on these conclusions a discussion is included in 6.2. From this discussion, recommendations for future research are given in 6.3.

6.1 Conclusions

The project goal was to eliminate the manual tasks in the data exchange between the platform for early payment and different ERP systems to decrease the time that is needed for the data exchange. For this, a suitable CDM to reduce the complexity of the data mapping is created by comparing a cross-industry standard with the schemas of the systems that had to be integrated. This CDM is used in an ESB to perform the process without any user involvement and reduced the time needed to perform the process to about one second.

In the current situation, the early payment process consists of four phases. In this process, the invoice must be uploaded manually, and the early payment results are added manually to the ERP system of the buyer. To eliminate these manual tasks and automate the data exchange between the different systems, the systems must be integrated at the data level.

By comparing four integration types, the Enterprise Application Integration type is identified as the best type to integrate multiple different systems. The Enterprise Service Bus can integrate systems with a very different data structure, by using a Canonical Data Model, which is understandable for each connected system.

In the process of creation, the human plays a key-role because common understanding is created in the human mind in an ad-hoc way. No structured method for the development of a suitable CDM to reduce the complexity of the data mapping in connecting the platform for early payment with different ERP systems was identified.

To validate whether an ESB with a suitable CDM influences the time needed for the data exchange, a prototype was created where one ERP system and the platform for early payment were connected to a CDM. This CDM was created based on an industry independent standard. With this prototype, the time needed for the data exchange was reduced to about 1 second whereas the data exchange of the current invoice upload is approximately 3 minutes and 19 seconds. From this test can be concluded that the time needed to exchange the data between ERP systems and the platform for early payment can be reduced significantly with the use of an Enterprise Service Bus and a Canonical Data Model that is based on an industry independent standard.

6.2 Discussion

In this paragraph, the implications of the conclusions and the limitations of the project and prototype will be discussed.

6.2.1 Project goal and research

The project goal was to decrease the time needed for the data exchange. For the measurement of the upload of invoices, only three people performed a baseline test. When more people would perform the baseline test, the results would become more conclusive. However, because the test was only used as an indication of the time that was needed to perform the current process, it is no problem that the measured time is not very precise. Measuring the time needed for the prototype of the invoice upload showed that this time is decreased tremendously. To enable more buyers to pay their invoices earlier using the platform for early payment, automating this data exchange seems to be a good investment.

Comparing the time needed for uploading of the run results from the early payment process into an ERP system was not possible since no information was available about the way these results are processed by the buyer into their ERP system. Further research is needed to find out if and how this connection can be automated and if the revenues outweigh the costs.

The situations for when an ESB or point-to-point integrations should be used are described in theory but in practice it also depends on the other factors like whether the connectors for the systems to the ESB already exist, if there is already a common understanding of the systems data structure and if the point-to-point integration has already been created for a similar system.

6.2.2 Prototype

The GS1 standard is a very substantial language with a lot of fields. These fields are not all included in the CDM of the prototype because most of them are rarely used and would make the CDM too big, what would have a negative effect on the overview of the data schema. However, the current selection contains a field that is not used yet. These fields are typed italic. A review is needed to sort out whether the right choices are made and whether these fields should stay in the CDM.

The API of the platform for early payment is still under development. For this project version, 1 of the API was used, but the API can change significantly because it is mostly based on the SAP ERP system. The encryption that is used in the API is currently under development by eMagiz, this makes it not possible at this moment to use this API.

The prototype is a Minimal Viable Product (MVP) where only the invoice upload is automated. The other types of data that must be exchanged can be created based on this invoice exchange because all data needed is already included in the CDM. Only the adapters and the mappings must be created.

The early payment results are not entered in the ERP because there are no standard adapters to write this data to the ERP. This adapter will have to be created by an expert of every ERP system. An adapter can then be reused for buyers who have the same ERP system.

In the MVP all invoices that are changed in the last 24 hours are sent every hour to the platform for early payment. No selection of invoices is made. When they are uploaded the duplicates are removed at the platform for early payment. This is a very inefficient approach since an invoice is uploaded multiple times, even when there is no change. To prevent this, a selection of the invoices that should be uploaded could be included in the prototype.

For the validation, every integration is different. Therefore, it is difficult to get a standard score on the time it takes to integrate every ERP system. This also depends on the user's knowledge about the ERP system, which can vary.

6.3 Recommendations

In a CDM a generic language is needed to flexibly manage the differences between the different standards and support the transformation between these standards. The language should be created during the development of the CDM. But because there is no structured method for the creation of a CDM, this language is developed in the human mind. This is a difficult process and it takes a lot of time to understand the semantics of all the standards that must be integrated. To overcome this a cross-industry standard like GS1 could be used in the development of a CDM, this standard also covers the semantics of the different standards and supports the developer in the creation of a CDM.

To create the mapping between the message of the ERP and the CDM, knowledge about the data fields in the database of the ERP system is required. This mapping can be created by an expert of the ERP, as the CDM itself is easy to understand since it is build using the GS1 standard. This mapping could then be reused for every buyer that uses the same ERP system.

Every time a new ERP system is added, it is important to check if the CDM covers all the data needed from and uploaded to that system. Also, when adding a field to the CDM, its name must be checked using the GS1 standard on readability to prevent that incomprehensible fields are added to the CDM.

6.4 Future research

In the research for a structured method to develop a CDM, no structured method that covered the integration of different standards was identified. To support the creation of a CDM, a structured method would decrease the time needed to develop a CDM and make the development process less intuition-based. Future research is needed to create a more structured approach for the development of a semantic and standard independent CDM.

Most ERP systems do not support dynamic discounting. This means buyers must find solutions on how to process the early payment results of the early payment process into their ERP system. More research is needed to find out what methods are used in practice and if this connection can be automated.

Bibliography

Chappell, D. A. (2004). *Enterprise service bus* (1st ed.). O'Reilly Media, Inc, Usa.

Chen, L. (2012). Integrating Cloud Computing Services Using Enterprise Service Bus (ESB). *Business and Management Research*, *1*(1), 26–31. https://doi.org/10.5430/bmr.v1n1p26

Dietrich, M., & Lemcke, J. (2011). A refined canonical data model for multi-schema integration and mapping. *Proceedings - 2011 8th IEEE International Conference on e-Business Engineering, ICEBE 2011*, 105–110. https://doi.org/10.1109/ICEBE.2011.26

Dietrich, M., Lemcke, J., & Stuhec, G. (2013). Semantic integration of e-business schemas via a canonical data model assessing the effort reduction for B2B message exchange. In *Proceedings - 2013 IEEE 10th International Conference on e-Business Engineering, ICEBE 2013* (pp. 106–111). Coventry: IEEE Computer Society. https://doi.org/10.1109/ICEBE.2013.16

Dietrich, M., Weissmann, D., Rech, J., & Stuhec, G. (2010). Multilingual extraction and mapping of dictionary entry names in business schema integration. In *iiWAS2010 - 12th International Conference on Information Integration and Web-Based Applications and Services* (pp. 863–866). Paris. https://doi.org/10.1145/1967486.1967635

Geevers, J. A. P. M. (2017, August). Het optimaliseren van het orderproces van CAPE Groep. Retrieved from http://essay.utwente.nl/73372/

Gelsomino, L., Mangiaracina, R., Perego, A., & Tumino, A. (2016). Supply Chain Finance: Modelling a Dynamic Discounting Programme. *Journal of Advanced Management Science*, *4*(4), 283–291. https://doi.org/10.12720/joams.4.4.283-291

GS1a. (2012). GS1 - The global language of Business. Retrieved July 5, 2018, from https://www.gs1.org/

Gulledge, T. (2006). What is integration? *Industrial Management & Data Systems*, *106*(1), 5–20. https://doi.org/10.1108/02635570610640979

Hohpe, G., & Woolf, B. (2004). *Enterprise integration patterns* (14th ed.). Boston: Addison-Wesley.

Land, R., & Crnkovic, I. (2003). Software systems integration and architectural analysis - A case study. In *INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, PROCEEDINGS* (pp. 338–347). https://doi.org/10.1109/ICSM.2003.1235441

Lee, J., Siau, K., & Hong, S. (2003). Enterprise integration with ERP and EAI. *Communications of the ACM*, *46*(2), 54–60. https://doi.org/10.1145/606272.606273

Lemcke, J., Stuhec, G., & Dietrich, M. (2012). Computing a canonical hierarchical schema. In *Proceedings of the I-ESA Conferences* (Vol. 5, pp. 305–315). https://doi.org/10.1007/978-1-4471-2819-9_27

Madhavan, J., Bernstein, P., Doan, A., & Halevy, A. (2005). Corpus-based Schema Matching, (Icde). Retrieved from https://pdfs.semanticscholar.org/ebbc/11cf9bcf5ed92111318504f5668c46060244.pdf

Netsuite | What is ERP (Enterprise Resource Planning)? (n.d.). Retrieved July 5, 2018, from http://www.netsuite.com/portal/resource/articles/erp/what-is-erp.shtml

Oracle | What is ERP? (n.d.). Retrieved July 5, 2018, from https://www.oracle.com/applications/erp/what-is-erp.html

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Perego, A., & Salgaro, A. (2010). Assessing the benefits of B2B trade cycle integration: A model in the home appliances industry. *Benchmarking*, *17*(4), 616–631. https://doi.org/10.1108/14635771011060611

Raap, W. B., Iacob, M.-E., van Sinderen, M., & Piest, S. (2016). An Architecture and Common Data Model for Open Data-Based Cargo-Tracking in Synchromodal Logistics. In Debruyne, C and Panetto, H and Meersman, R and Dillon, T and Kuhn, E and OSullivan, D and Ardagna, CA (Ed.), *ON THE MOVE TO MEANINGFUL INTERNET SYSTEMS: OTM 2016 CONFERENCES* (Vol. 10033, pp. 327–343). GEWERBESTRASSE 11, CHAM, CH-6330, SWITZERLAND: SPRINGER INT PUBLISHING AG. https://doi.org/10.1007/978-3-319-48472-3_19

Roman, D. (2006). Canonical data & process models for B2B integration. In *CEUR Workshop Proceedings* (Vol. 170). Seoul. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-84884301188&partnerID=40&md5=4da82506cd23877266f1ad30eed3ac64

Smith, A., & McBrien, P. (2006). Inter model data exchange of type information via a common type hierarchy. In *CEUR Workshop Proceedings* (Vol. 238, pp. 307–321). Luxemburg. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-84875944694&partnerID=40&md5=441d2bdb5b35d14f3f9346aa819475a8

Sutherland, J. V., Solingen, R. van., & Rustenburg, E. (2011). *The power of Scrum*. CreateSpace.

Taibi, D., & Lenarduzzi, V. (2016). MVP explained: A systematic mapping on the definition of minimum viable product. *42Th Euromicro Conference on Software Engineering and Advanced Applications 2016, Cyprus*.

Teusch, A. (2014). Design of hierarchically structured canonical data models for Very Large Business Applications (VLBA) [Entwurf von hierarchisch-strukturierten kanonischen Datenmodellen für VLBA]. In S. L. Kundisch D. Beckmann L. (Ed.), *Tagungsband Multikonferenz Wirtschaftsinformatik 2014, MKWI 2014* (pp. 2257–2270). University of Paderborn. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-84984706274&partnerID=40&md5=5fa3db85294e8fb94b9c21bf7ee75afa

Weske, M. (2012). *Business Process Management*. Springer. https://doi.org/10.1007/978-3-642-28616-2

Appendix

A. Literature review protocol

Search string	Scope	Date of search	Date range	Nr. Of entries	Delta
Search protocol for Scopus					
"Common data model" and "design"	Title, keywords and abstract	12-4-2018	all	103	
"Canonical data model" and "design"	Title, keywords and abstract	12-4-2018	all	5	
"Common data model" and "development"	Title, keywords and abstract	12-4-2018	all	110	
"Canonical data model" and "development"	Title, keywords and abstract	12-4-2018	all	8	
Total in Mendeley				226	
Removing duplicates				180	-46
Selecting based on inclusion/exclusion criteria				49	-131
Removed after abstract				33	-16
Based on availability				25	-8
After reading				10	-15
Included				3	+2
Total selected for review				12	

Include if…	***Because…***
The paper describes the requirements of a data model	*These requirements can be tested in the final model*
The paper describes the process of development of the data model	*This helps with defining a development method.*
The paper describes a modeling notation	*Modeling notations must be compared.*
The paper is freely accessible via the university	*There is no research budget.*
Exclude if…	***Because…***
The paper is about healthcare	*The study is not about a new solution for patient data*
The paper is about geospatial data	*This study is about a different sector*
The paper is about the data of a production or manufacturing process (CAD/CAM), location data, vocabulary or a simulation.	*This is about a different sector*
An existing model is improved, or the application of the model is described	*This study is about the development methods, not about the model itself*
The paper is about database design	*This study is about a method to integrate existing systems, not about the creation of a new multi database*

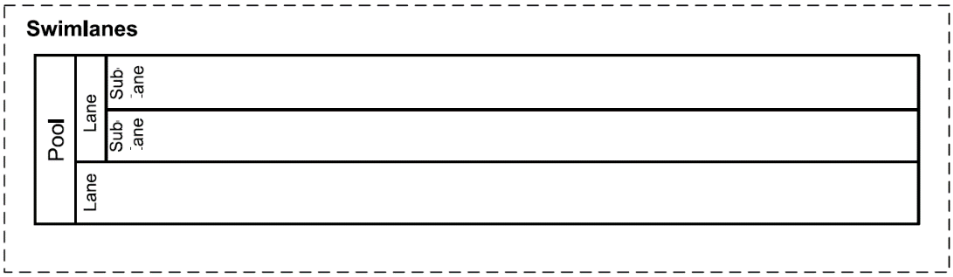
Literature results

Title	Authors	Topic
A formalization of semantic schema integration	McBrien, P Poulovassilis, A	Transformation
A refined canonical data model for multi-schema integration and mapping	Dietrich, Michael Lemcke, Jens	Schema correspondence
A Self-Organizing Knowledge Representation Scheme for Extensible Heterogeneous Information Environment	Sull, W Kashyap, R L	Translation
Canonical data & process models for B2B integration	Roman, D	Matching approach
Computing a Canonical Hierarchical Schema	Lemcke, Jens, Stuhec, Gunther Dietrich, Michael	Algorithmic approach
Integration and evolution of XML data via common data model	Klímek, J Nečaský, M	Different views of schemas
Intermodel data exchange of type information via a common type hierarchy	Smith, A McBrien, P	Type information
Multilingual extraction and mapping of dictionary entry names in business schema integration	Dietrich, M, Weissmann, D Rech, J, Stuhec, G	DEN for schema mapping
Semantic integration of e-business schemas via a canonical data model assessing the effort reduction for B2B message exchange	Dietrich, Michael Lemcke, Jens, Stuhec, Gunther	Standards fields
Software Systems Integration and Architectural Analysis - A Case Study	Land, R Crnkovic, I	Levels of integrations
Utility applications should be integrated with an interface based on a canonical data model, not directly with each other	Robinson, G Zhou, M J	Mappings with a CDM
Enterprise integration patterns	Hohpe, Gregor Woolf, Bobby	Integration of systems

B. BPMN and ArchiMate

BPMN

The Business Process Modelling Notation is a standard notation that can be used to display business processes in a graphically. The modeling notation uses the Business Process Diagram and is based on the techniques that are also used in flowcharts. Below the different elements that are used within the notation are explained (Weske, 2012).



Swimlanes are used to divide the different roles and organizations that are involved in the process. A pool is a participant in the process and the lane can be used to specify a role with the company or a person.

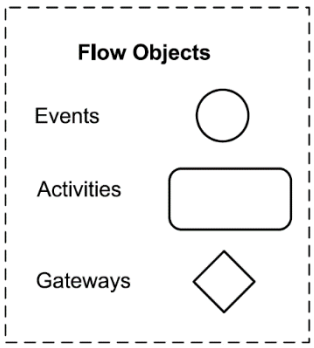


Figure 12

Flow objects (figure 12)
An event is something that happens during the business process, for example, the receiving of an invoice.
An activity is a task that is performed within an organization.
The gateway is used to model a decision that is made during the process.

Figure 11

Event and activity tasks (figure 11)
Events and activities can have different tasks. A send and receive are tasks that involve sending or receive something. The user task involves user interaction. The manual tasks are performed without the support of software systems. The business rule task triggers a business rule. The service task is implemented by a piece of software. The script task uses a scripting language to be performed.

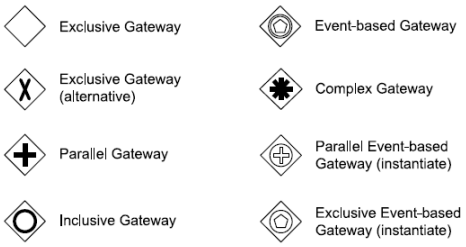


Figure 13

Gateways (figure 13)
There are also different types of gateways. The gateways can be used as a join of split node. The gateways can have multiple incoming or outgoing edges. With an exclusive gateway, one flow is selected. The parallel gateway makes multiple flows at the same time possible. The inclusive gateway is used when one must be chosen. The event gateway is triggered upon an event and a complex gateway combines a split and a join.

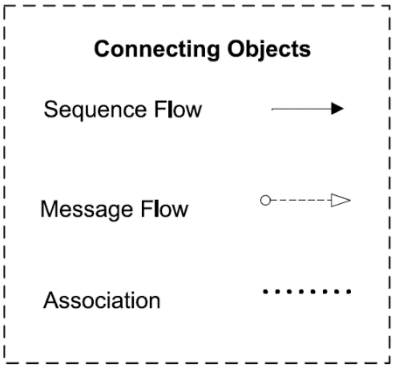


Figure 14

Connecting objects (figure 14)
The sequence flow specifies the order in which the activities are performed.
The message flow shows the messages that are sent between the different participants of the process.
An association is used to show the link with an artifact.

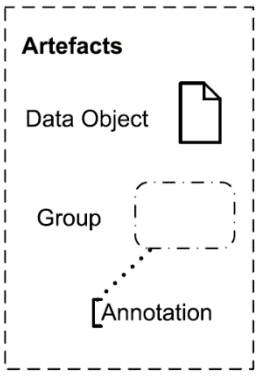


Figure 15

Artifacts (figure 15)
A data object is used to show the data that is used or produced by an activity.
A group can be used to show a part of the process that is a collection of a set of objects.
An annotation can be used to explained parts of the process.

ArchiMate

Archimate is a standard that offers a language for representing and describing different architectural layers. Archimate is a modeling language that provides a uniform representation of the different domains within a company and their underlying relations and dependencies.
The Archimate modeling language distinguishes between three layers: The business layer, where a description of the structure and interaction between the business strategy, organization, functions, and business processes. The application layer supports the business layer with the services that must be realized by providing applications. This is used to model the information systems and their architecture that describes the structure and interaction between the applications. And the technology layer provides storage and communication that is needed for the applications, this shows how the technology support the applications (Geevers, 2017).
The archimate language doesn't provide the level of detail that other languages like BPMN

and UML provide but it creates an abstraction.
What is similar with these higher detailed design languages is that Archimate does provide a standard graphical notation.

C. Explanation of Terms

Here the terms that are used throughout the thesis are explained.

ERP system

ERP stands for Enterprise Resource Planning. These are systems that are used to manage the business processes of accounting, procurement, project management and manufacturing in one system. The ERP started as an extension of the Material Requirements Planning and computer integrated manufacturing. The systems later extended with finance and accounting, maintenance, and HRM and extended to all (core) enterprise functions. The system facilitates flow between the business functions as it integrates all functions into a complete system to streamline the processes and information ("Oracle | What is ERP?" n.d.).
Different functions used to have different systems which host related data, one object could be stored in different databases. The link between this data was made with dedicated links. Because of the multiple data dependencies, it is difficult to change a relational system. A change had to be reflected in multiple systems, the lack of integration between these systems often lead to data inconsistency.
ERP systems were developed to tackle this problem, they have an integrated database for all the systems that are used within an organization. This ensures that the database is consisted, in an ERP a set of application modules provides desired functionality for different business processes.
Because the ERP system shares data, it eliminates the duplication and provides a single source of truth. A key principle of ERP is that centralizes the collection of data, then the data is correct, up to date and complete ("Netsuite | What is ERP (Enterprise Resource Planning)?" n.d.).

D. Baseline measurement invoice upload

The goal of this test is to measure the duration of the whole process of selecting an invoice, creating a file for upload and uploading the file to the platform for early payment This measurement will be used as a baseline measurement of the current manual process of the invoice upload.

The invoice upload process consists of 4 phases:

1. Select the data in Exact
2. Enter the data into a CSV file
3. Make sure the format is correct
4. Login on the platform for early payment and upload the file to the platform for early payment

Phase 1

Starting the ERP system and selecting the invoice in the Exact.

Start a stopwatch

Step 1: Open Exact
Step 2: Go to factuur (invoice) see
Step 3: Go to factuurhistorie (invoice history)
Step 4: Select a random invoice

Stop the stopwatch.

Time of phase 1 was:

Reset the stopwatch

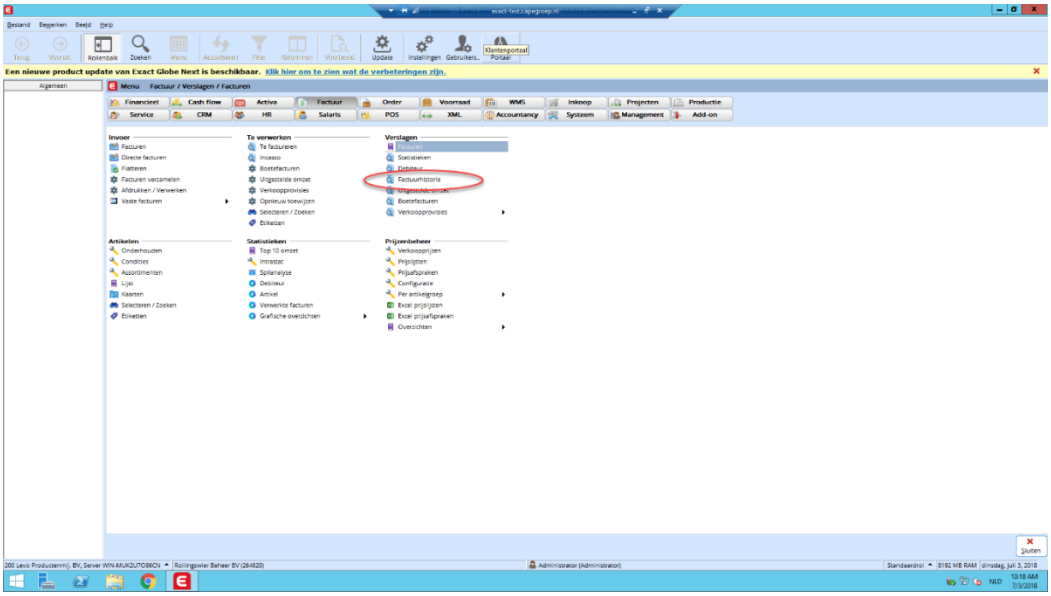


Figure 16

Phase 2

Entering the invoice data into a CSV file.

Start a stopwatch

Step 1: Open the invoice
Step 2: Enter the following data into a CSV file, see figure 16 for the location of the data in Exact:

1. Invoice reference or number that you use to track this invoice in your own system (e.g.: "VOORB2123332")
2. Supplier code which you use to refer to a specific supplier in your system (e.g.: "EXCOMP23" for "Example Company 23 Ltd.")
3. currency ("EUR")

4. total invoice amount, excluding VAT (e.g.: "112,04" > our system recognizes the number of decimals and format automatically)
5. total invoice amount, including VAT factuurbedrag, inclusief BTW (e.g.: "112,04" > our system recognizes the number of decimals and format automatically)
6. Invoice date using the format DD-MM-JJ or DD/MM/JJ (including the separator). Our system automatically detects the if you use a 2 or 4-digit year (e.g. "31-10-2011" of 31/10/11")
7. due date of the or regular payment term (our system recognizes both, so.:"31-12-2018" or "45")
8. invoice reference or number as indicated by your suppliers on the invoice (e.g.: "123332") (Preferably without supplier code)
9. If you use a so-called 'blocked account number': the partial amount destined for this account (e.g.: "34,24"). If there's no 'blocked account number' space or a 0 (zero)

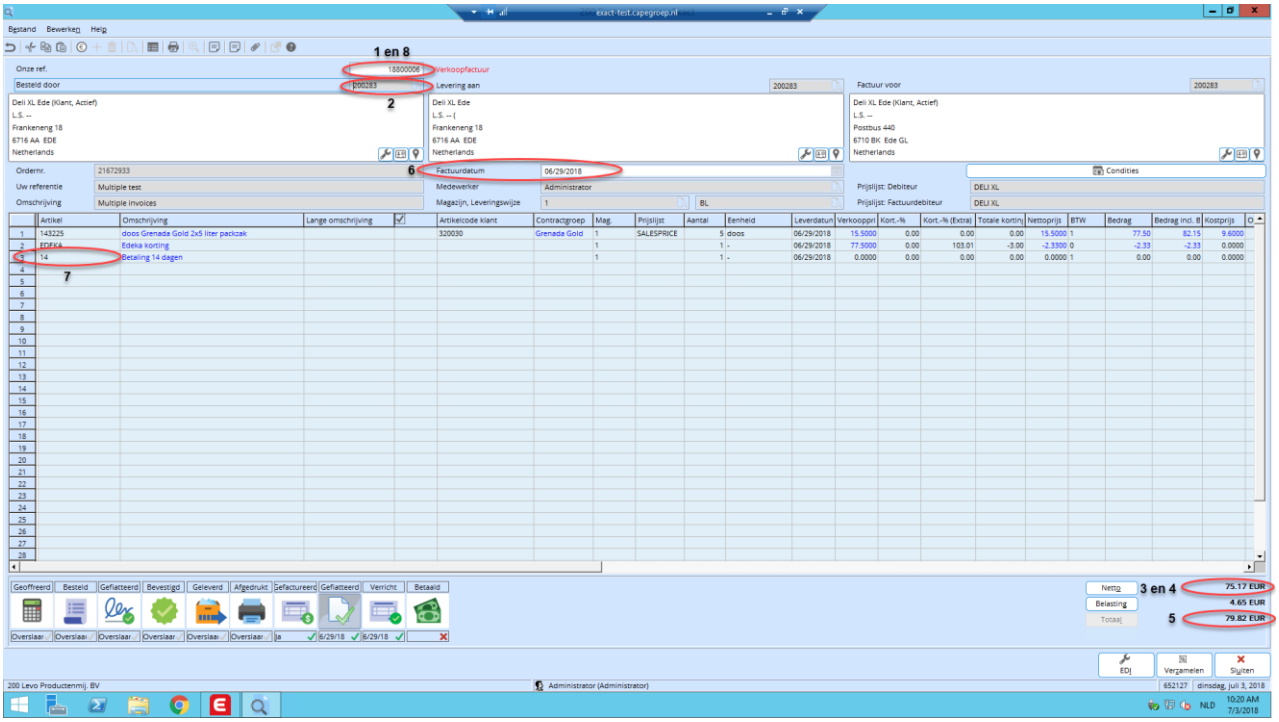


Figure 17

followed by the separator ';' (semicolon)

Stop the stopwatch.

Time of phase 2 was:

Reset the stopwatch

Phase 3

Start a stopwatch

Step 1: Verify the correctness of the format of the data entered.

Example:

1. "2123332"
2. "218492"
3. ISO currency code ("EUR")
4. Number separated by comma (e.g.: "134,45")
5. Number separated by comma (e.g.: "134,45")
6. DD-MM-JJ or DD/MM/JJ e.g. ("31-10-2011" or "31/10/11")
7. DD-MM-JJ or DD/MM/JJ or number of days (e.g.:"31-12-2011" or "45")
8. E.g.: "2123332"
9. 0

Step 2: Save the file as a Comma Separated File (.csv)

Stop the stopwatch.

Time of phase 3 was:

Reset the stopwatch

Phase 4

Upload the created file to the platform for early payment

Start a stopwatch

Step 1: Log in on the platform for early payment
Step 2: Go to data upload
Step 3: Upload the created CSV file from the stored location
Step 4: Wait until the file is uploaded. If there are no errors, the invoice is completed.

Stop the stopwatch.

Time of phase 4 was:

Total time:

E. Prototype

Depth	Name	Story Type	Story Points	Status
Sprint	Capture and Design phase			
+	Add the platform for early payment system	Feature		To-do
+	Add Exact system	Feature	8	To-do
+	Define message type invoice	Feature		To-do
+	Define message type Run Results	Feature		To-do
+	Define message type Remittance advice	Feature		To-do
+	Create system message Run Results for the platform for early payment	Feature	2	To-do
+	Create system message invoice Exact	Feature	3	To-do
+	Define CDM entity Vendor	Feature	13	To-do
++	Improve by comparing to GS1	Feature		To-do
++	Improve by comparing to Navision	Feature		To-do
++	Improve by comparing to Exact	Feature		To-do
+	Define CDM entity Invoice	Feature	5	To-do
++	Improve by comparing to GS1	Feature		To-do
++	Improve by comparing to Navision	Feature		To-do
++	Improve by comparing to Exact	Feature		To-do
++	Add multiple invoices	Feature		To-do
++	Add invoice lines	Feature		To-do
+	Define CDM entity Payment term	Feature	1	To-do
++	Improve by comparing to GS1	Feature		To-do
++	Improve by comparing to Navision	Feature		To-do
++	Improve by comparing to Exact	Feature		To-do
+	Define CDM entity Payment method	Feature	1	To-do
++	Improve by comparing to GS1	Feature		To-do
++	Improve by comparing to Navision	Feature		To-do
++	Improve by comparing to Exact	Feature		To-do
+	Define CDM entity Payment confirmation	Feature	1	To-do
++	Improve by comparing to GS1	Feature		To-do
++	Improve by comparing to Navision	Feature		To-do
++	Improve by comparing to Exact	Feature		To-do
+	Define CDM entity Run results	Feature	3	To-do
++	Improve by comparing to GS1	Feature		To-do
++	Improve by comparing to Navision	Feature		To-do
++	Improve by comparing to Exact	Feature		To-do
+	Message mapping invoice Exact --> CDM	Feature		To-do
+	Message mapping invoice CDM --> platform for early payment	Feature		To-do
++	Multiple invoices	Feature		To-do
+	Message mapping run results Platform for early payment --> CDM	Feature	3	To-do
Sprint	Create phase			
+	Build connector invoices Exact	Feature		To-do
+	Build onramp invoices Exact to CDM	Feature	5	To-do
+	Build routing of Invoice	Feature	5	To-do
+	Build offramp CDM to the Platform for early payment	Feature		To-do
+	Build onramp run results from platform for early payment to CDM	Feature		To-do
+	Build connector run results platform for early payment	Feature		To-do
+	Build routing run results	Feature		To-do

Figures of the prototype

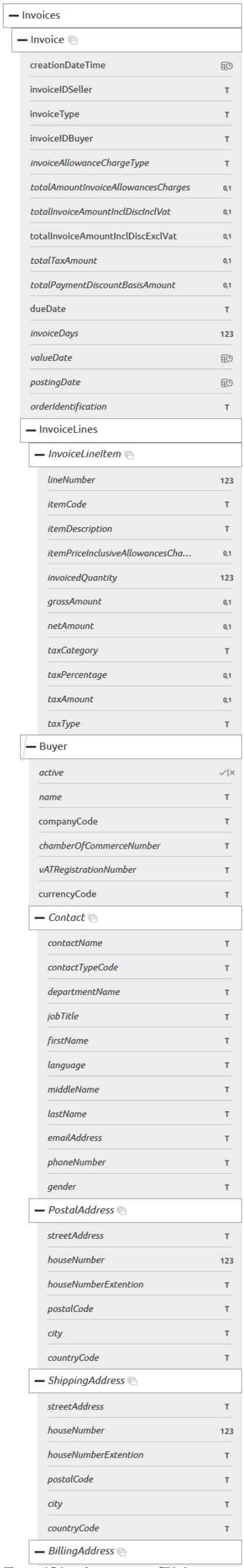


Figure 19 Invoice message CDM

Figure 18 Invoice message CDM

- GetInvoiceHistoryObjResponse
  - GetInvoiceHistoryObjResult
    - ErrorMsg T
    - Success T
    - Invoices
      - InvoiceHistoryFull
        - Entity
          - *AdditionalDateField01* T
          - *AdditionalDateField02* T
          - *AdditionalDateField03* T
          - *AdditionalDateField04* T
          - *AdditionalDateField05* T
          - *AdditionalField01* T
          - *AdditionalField02* T
          - *AdditionalField03* T
          - *AdditionalField04* T
          - *AdditionalField05* T
          - *AdditionalField06* T
          - *AdditionalField07* T
          - *AdditionalField08* T
          - *AdditionalField09* T
          - *AdditionalField10* T
          - *AdditionalFieldsStatus* T
          - *AdditionalNumberField01* T
          - *AdditionalNumberField02* T
          - *AdditionalNumberField03* T
          - *AdditionalNumberField04* T
          - *AdditionalNumberField05* T
          - *AdditionalNumberField06* T
          - *AdditionalNumberField07* T
          - *AdditionalNumberField08* T
          - *AdditionalNumberField09* T
          - *AdditionalNumberField10* T
          - *AdditionalYesNoField01* T
          - *AdditionalYesNoField02* T
          - *AdditionalYesNoField03* T
          - *AdditionalYesNoField04* T
          - *AdditionalYesNoField05* T
          - CurrencyCode T
          - CurrencyRate 0,1
          - DeliveryDate
          - DeliveryDebtorAddress T
          - DeliveryDebtorCity T
          - DeliveryDebtorName T
          - DeliveryDebtorPostCode T
          - *DeliveryNoteNumber* 123
          - ID 123
          - InvoiceDate
          - InvoiceDebtor 123
          - InvoiceDebtorAddress1 T
          - InvoiceDebtorCity T
          - InvoiceDebtorContactPerson T
          - InvoiceDebtorCountry T
          - InvoiceDebtorName T
          - InvoiceDebtorPostCode T
          - InvoiceNumber 123
          - InvoiceType T
          - JournalCode 123
          - OrderDate
          - OrderDebtor T
          - OrderDebtorAddress1 T
          - OrderDebtorCity T
          - OrderDebtorContactPerson T
          - OrderDebtorCountry T
          - OrderDebtorPostCode T
          - PaymentCondition 123
          - SalesOrderNumber 123
          - SalesPrice 0,1
          - *VATNumber* T
          - YourReference T
          - DateDue
          - InvoiceAmountDue 0,1
          - IsOpen T
          - SysGuid T
        - InvoiceLines
          - InvoiceHistoryLineExtra

Figure 21 Exact system message

- InvoiceLines
  - InvoiceHistoryLineExtra
    - *AdditionalDateField01* T
    - *AdditionalDateField02* T
    - *AdditionalDateField03* T
    - *AdditionalDateField04* T
    - *AdditionalDateField05* T
    - *AdditionalField01* T
    - *AdditionalField02* T
    - *AdditionalField03* T
    - *AdditionalField04* T
    - *AdditionalField05* T
    - *AdditionalField06* T
    - *AdditionalField07* T
    - *AdditionalField08* T
    - *AdditionalField09* T
    - *AdditionalField10* T
    - *AdditionalFieldsStatus* T
    - *AdditionalNumberField01* T
    - *AdditionalNumberField02* T
    - *AdditionalNumberField03* T
    - *AdditionalNumberField04* T
    - *AdditionalNumberField05* T
    - *AdditionalNumberField06* T
    - *AdditionalNumberField07* T
    - *AdditionalNumberField08* T
    - *AdditionalNumberField09* T
    - *AdditionalNumberField10* T
    - *AdditionalYesNoField01* T
    - *AdditionalYesNoField02* T
    - *AdditionalYesNoField03* T
    - *AdditionalYesNoField04* T
    - *AdditionalYesNoField05* T
    - AmountInclDiscExclVAT 0,1
    - DeliveryDate
    - DiscountPercentage 0,1
    - ID 123
    - InvoiceDate
    - InvoiceNumber T
    - *ItemCode* T
    - *ItemDescription* T
    - JournalCode 123
    - LineNumber 123
    - NetPrice 0,1
    - *PriceList* T
    - Quantity 123
    - SalesOrderNumber 123
    - SalesPrice 0,1
    - *SerialNumber* T
    - TaxCode 123
    - SysGuid T

Figure 20 Exact system message

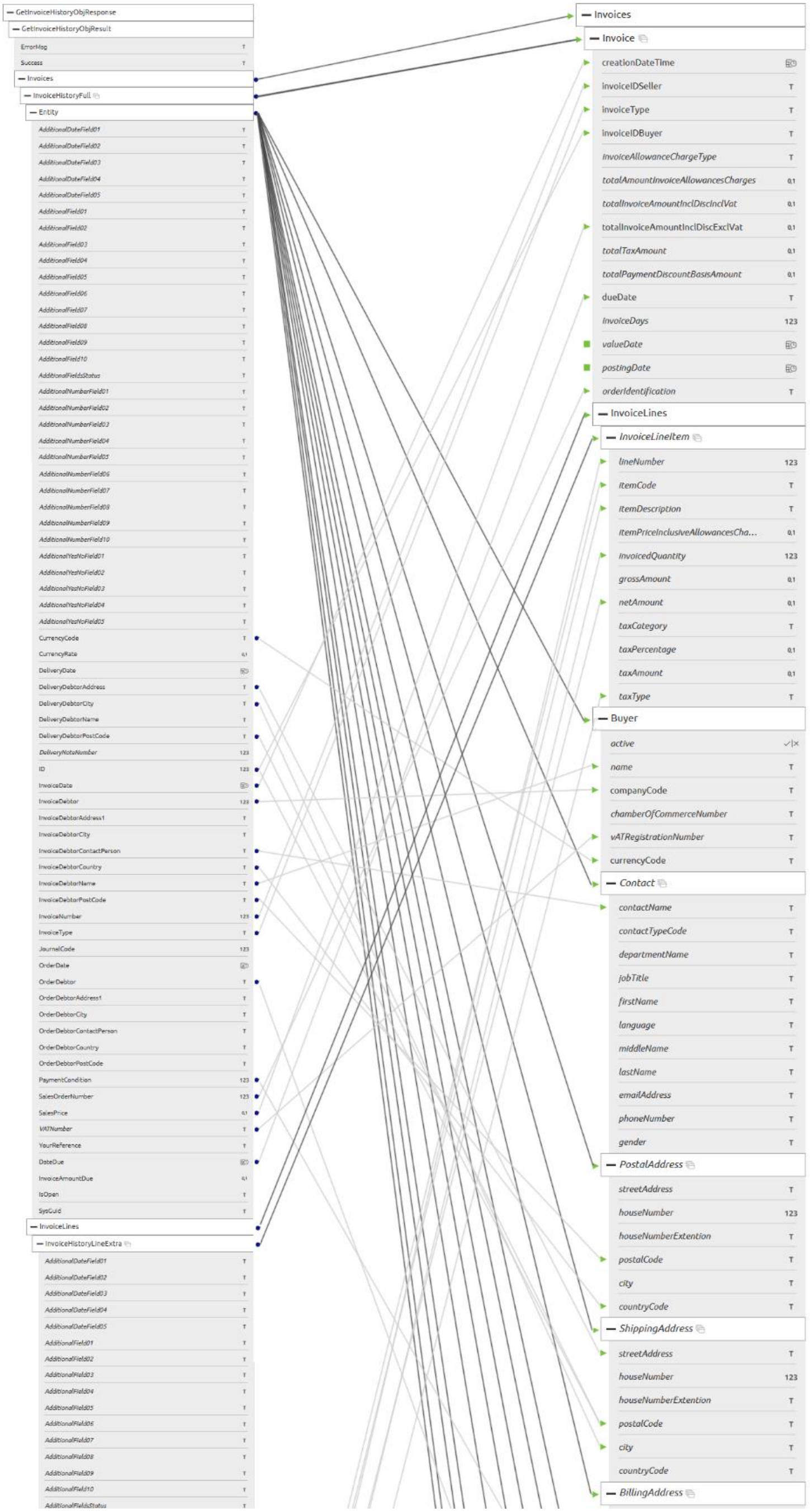


Figure 22 Mapping Exact to CDM

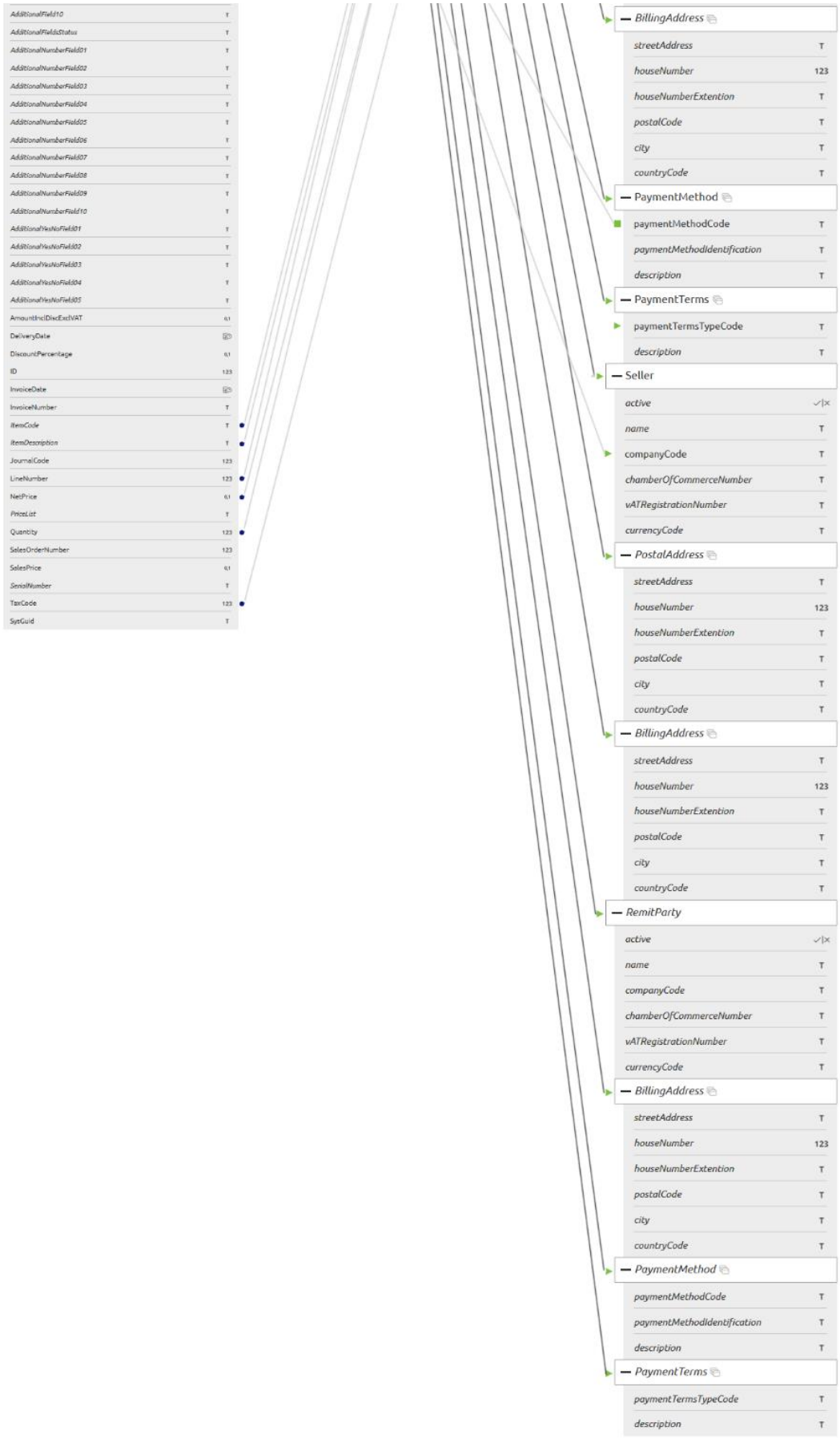


Figure 23 Mapping Exact to CDM

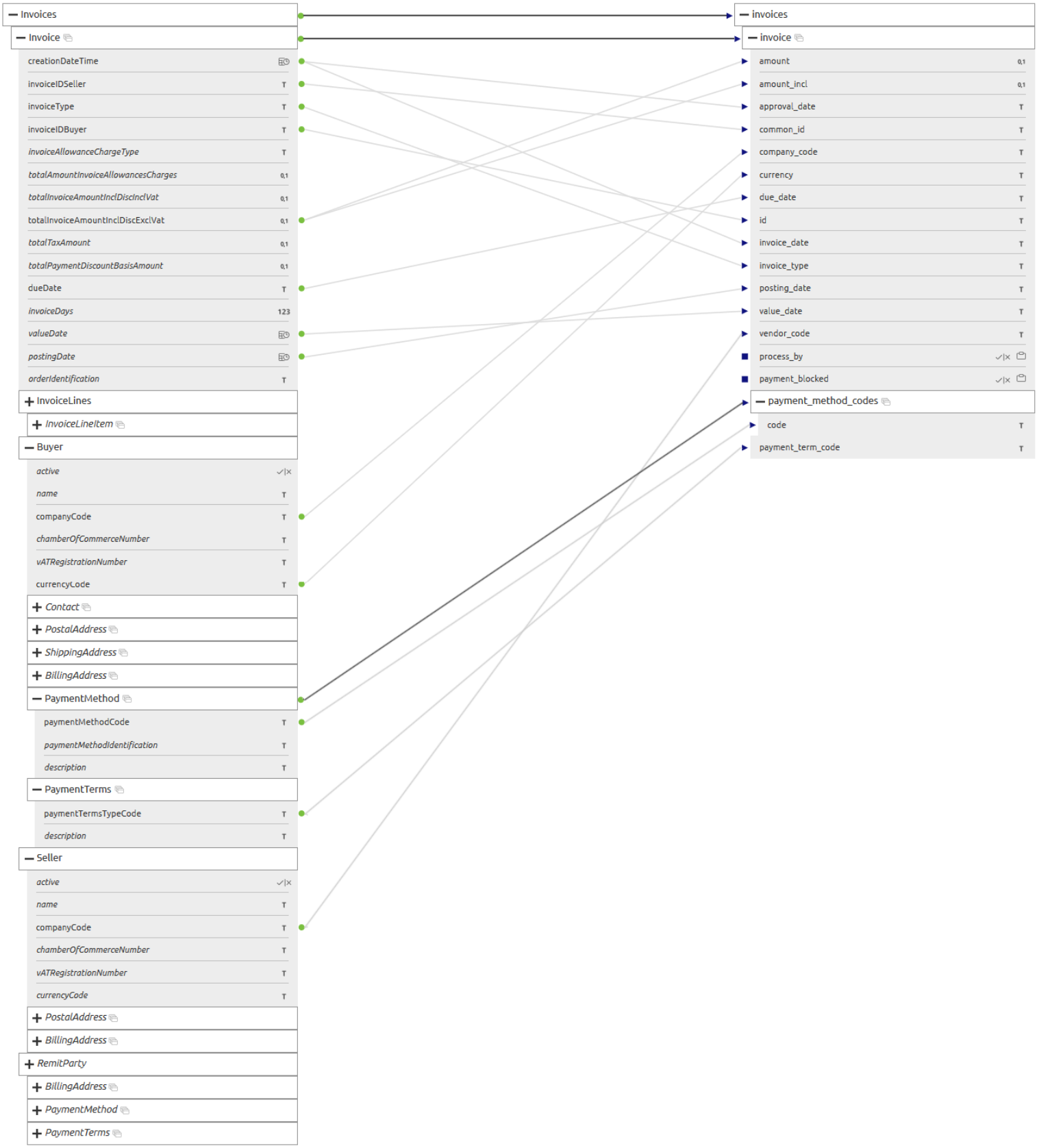


Figure 24 Mapping CDM to Platform for early payment

```xml
<?xml version="1.0"?>
<invoices>
   <invoice>
      <amount>297.67</amount>
      <amount_incl>297.67</amount_incl>
      <approval_date>2018-06-29T00:00:00</approval_date>
      <common_id>18800005</common_id>
      <company_code>123456</company_code>
      <currency>EUR</currency>
      <due_date>2018-07-13T00:00:00</due_date>
      <id>17478</id>
      <invoice_date>2018-06-29T00:00:00</invoice_date>
      <invoice_type>V</invoice_type>
      <posting_date>2018-06-29T13:09:03.632+02:00</posting_date>
      <value_date>2018-06-29T13:09:03.632+02:00</value_date>
      <vendor_code>DXLE</vendor_code>
      <process_by_>true</process_by_>
      <payment_blocked>false</payment_blocked>
      <payment_method_codes>
         <code>20</code>
      </payment_method_codes>
      <payment_term_code>14</payment_term_code>
   </invoice>
   <invoice>
      <amount>79.82</amount>
      <amount_incl>79.82</amount_incl>
      <approval_date>2018-06-29T00:00:00</approval_date>
      <common_id>18800006</common_id>
      <company_code>123456</company_code>
      <currency>EUR</currency>
      <due_date>2018-07-13T00:00:00</due_date>
      <id>17479</id>
      <invoice_date>2018-06-29T00:00:00</invoice_date>
      <invoice_type>V</invoice_type>
      <posting_date>2018-06-29T13:09:03.632+02:00</posting_date>
      <value_date>2018-06-29T13:09:03.632+02:00</value_date>
      <vendor_code>DXLE</vendor_code>
      <process_by_>true</process_by_>
      <payment_blocked>false</payment_blocked>
      <payment_method_codes>
         <code>20</code>
      </payment_method_codes>
      <payment_term_code>14</payment_term_code>
   </invoice>
</invoices>
```

Figure 25 Invoices for the Platform for early payment