Defining Electronic Data Interchange Transactions with UML

Christian Huemer
Institute of Applied Computer Science and Information Systems, University of Vienna,
Liebiggasse 4/3-4, A-1010 Vienna, Austria
ch@ifs.univie.ac.at

Abstract. Current EDI standards include a lot of complexity and their integration into existing applications is extremely expensive. This is due to the fact that current EDI standard messages are based on data schemes intended to capture all data that may appear in any business scenario of the corresponding business transaction. The standards do not capture the business requirements and scenarios that lead to the hierarchical EDI message design. Considering the requirements of modelling inter-organizational business transactions we present concepts based on Unified Process and UML to support modelling EDI scenarios. Resulting UML diagrams can support the design of current EDI messages as well as the design of future EDI standards based on object technology.

1 Introduction

Electronic Data Interchange (EDI) is the application-to-application exchange of business-related data based on a format understood by both (all) trading partners using an electronic transmission medium in order to carry out a business transaction [8]. The economic advantages of electronic data interchange (EDI) are widely recognized in the business-to-business area of electronic commerce. Nevertheless, the number of organizations and companies employing EDI is relatively small compared to the total number of businesses worldwide. The huge difference is caused by the fact that current EDI standards, like UN/EDIFACT [1] or ANSI X12 [6], include a lot of complexity and their integration into existing applications is extremely expensive.

Each EDI standard message is based on a data model for a single business transaction. It is created by volunteers from the business world working in the standardization bodies, who put their business sector know-how into a data schema which is written down in the EDI standard syntax [16]. As a result an EDI standard message is a data schema that is intended to capture all data that may appear in any business document of the corresponding business transaction. But message developers do not use a well defined method to collect and structure the user requirements. In absence of documentation on user requirements, the only output of the standardization process is the EDI message structure itself. Without any documentation it is difficult to keep track of why a certain message component has been introduced.

Accordingly, the standards are not tied back to the business process as a whole. As a consequence, people not directly involved in message development, may not understand the complexity included in the messages. This is made worse by the fact that standard messages include optionality without explaining under which conditions these options are to be used. Furthermore, the same information can be passed in different ways within a standard message.

Before starting an EDI interchange, the involved partners have to trim down the EDI standard messages to suite their requirements and to specify the semantics with almost no optionality within message implementation guidelines. Usually, they interpret the standard message structure in their own way, which might be quite different to the

standard's intention. Hence, MIGs for different partnerships usually stay in conflict. It follows that business partners - although using a so-called standard message - in fact, use different corresponding proprietary messages for different business relationships.

To overcome this unlucky situation of semantic ambiguities and different interpretations of a 'standard', it is a prerequisite to define semantically complete and unambiguous data models for an EDI business transaction.

Therefore, EDI standard development should start with a careful analysis of the EDI business transaction in question. In this starting phase the techniques of business modelling will ensure the development of 'standardized' EDI business transactions. According to the Open-edi reference model (ISO standard 14662) [11] this allows a separation of the semantics of a business process and the representation of data in a transfer syntax. The business models present the BOV (business oriented view) standards of Open-edi, which describe the business needs of an EDI transaction. BOV related standards provide the tools for formal business description(s) of the external behaviour of organizations, as seen by other organizations, in view of achieving a business goal. As such, the BOV related standards provide a means for capturing the static and dynamic requirements of the real world. The BOV standards are not influenced by the FSV (functional service view) standards, which address the supporting services meeting the mechanistic needs of Open-edi. The separation of BOV and FSV provides two main advantages compared to the current approach. Firstly, the BOV standards document the business requirements and allow for a semantically consistent definition of an EDI transaction. Secondly, the BOV standards will remain stable even if different EDI methodologies in the FSV are used. Therefore BOV standards can serve as basis for the development of current EDI messages in EDIFACT or X12 syntax, of XML/ EDI interchanges, as well as for future EDI standards based on object-oriented technology.

According to the above mentioned advantages the UN/CEFACT Steering Group proposed the following resolution for adoption by the UN/EDIFACT Working Group (EWG) in January 1998: 'The UN/CEFACT Steering Group (CSG) resolves that business and information modelling is an essential requirement to the future of UN/EDIFACT and that the implementation of business and information modelling is a critical objective of the CEFACT strategy and its attendant work program' [3].

The Techniques and Methodology Working Group (TMWG) of UN/CEFACT investigated in three different modelling techniques - namely IDEF [22], EXPRESS-G [23] and UML [21] - as possible candidates in an EDI environment. According to a careful analysis of these techniques TMWG selected UML as the technique for UN/CEFACT use in business process and information modelling.

In a current project TMWG is developing the OO-edi standard [20], which is a BOV related standard on the basis of UML models. Since UML is 'only' a modelling language, a methodology for applying UML modelling techniques within the OO-edi standards development process is needed [4, 7]. Therefore, TMWG has selected the Unified Process [13] as candidate process to start with. Nevertheless, the Unified Process has to be adapted to the specific needs of modelling EDI transactions. Furthermore, within Unified Process exact guidelines on how to exactly use the UML models must be specified. The core of this paper will focus on concepts have to be used when modelling EDI transactions with UML.

Since UML has been designed to support the software development process, it is our goal to gain experience on the suitability of UML to support business modelling [9,14] with a special focus on inter-organizational business modelling. This paper presents the experience gained from a first demonstration project. Nevertheless, using UML for

business modelling as well as for a following software process to design off-the shelf EDI software would eliminate a paradigm shift necessary when using another business modelling technique. Consequently, an UML-based methodology will ensure a consistent overall design process.

The remainder of this paper is structured as follows. In Section 2 we introduce the specific needs of BOV related standards on a business modelling methodology. How these requirements can be captured within Unified Process is described in Section 3. We conclude with a short summary on the experience gained form a demonstration project.

2 Requirements on UML for Modelling EDI Transactions

The development of EDI standards requires a full understanding of the problem domain. In an EDI environment this problem domain is usually an inter-organizational system. A business modelling methodology should be used to ensure a better understanding of the inter-organizational system [12]. An UML-based methodology should help to visualize an inter-organizational system and permit the specification of its structure and behaviour.

In order to specify a UML-based methodology the first task is to define the expected results to be delivered by the models. TMWG takes the view that the BOV standards and consequently the relating business models must cover a definition of the business domain to be supported, a specification of the requirements on the EDI transaction and a specification of the common EDI business objects (data structures and their relation to processes using the data) [5]. Accordingly, the current focus is not on software development, but on using UML to describe business transactions on a conceptual level. As a consequence, the proposed methodology concentrates on the following core workflows of the Unified Process [13]: the business modelling workflow, the requirements workflow and the analysis and design workflow.

The OO-edi approach does not take advantage of the implementation and following workflows. This is due to the fact that the implementation workflow will go beyond the goals of BOV standards, because it has to consider a concrete transfer mechanism which is part of the FSV layer of Open-edi standards. Only at a later stage, once the benefits of OO-edi have been tested, it will be possible to take the additional steps of working more closely with service and software providers to finally implement off-the-shelf EDI software.

Therefore, the current focus is on providing modelling guidelines for the starting three core workflows. In order to verify the suitability of UML for modelling EDI transactions the following requirements on a methodology to support inter-organizational business modelling have been identified [10]:

- The methodology must ensure that all involved organizations have a common understanding on the problem domain. Therefore, the business terms used in the description of the problem domain must be ‘agreed’ upon to ensure a semantically identical understanding among the involved organizations.
- The boundaries of the inter-organizational business system must be well understood. It must be clear what is inside the scope of the business transaction and what is outside.
- Since the focus is on the inter-organizational system the functions internally performed by the information systems of the involved organizations are not explicitly part of the system to be modelled. Nevertheless, the interfaces to these information systems have to be clearly identified. The inter-organizational system must define

the functions expected to be fulfilled by internal information systems.

- The business transaction to be focused within the models must be defined to an extend that avoids a proliferation of design models. For example to model all the transactions in an international trade transaction would result in a multitude of design models which cannot be traced efficiently. Therefore, the models have to focus on subtransactions which are by itself meaningful (e.g., order process). Consequently, the models of a considered business transaction must enable links to concepts detailed in another business transaction. Furthermore, the interfaces to a system modelled in another business transaction must be clearly indentifiable.
- It should be easy to distinguish between interfaces to the internal information systems (which support the core functions of the business transaction) and interfaces to external systems (where details are documented in another business transaction).
- The sequence of activities to be performed by each party in the business transaction should be clearly identified. In particular, these activities that lead to different scenarios must be expressed in the models. An 'easy to use' method should ensure that business experts without modelling experience can deliver input to design their organizations' business practices and express their requirements on services from partners.
- Services provided by each organization to contribute to the business transaction must be defined. It must be clear what an organization expects as input to perform a service and what the organization returns as output to the requester of a service.
- Models must be able to capture the communication processes between the organizations. The order of the communication processes must be defined. Accordingly the preconditions to be reached that a certain process can start must be declared.
- Data structures supporting the information flows must be identified. It must be clear which components of the data structure must be instantiated in an interchange and which instances are optional. Furthermore, rules for the instantiation should be identified.
- The guidelines must support the modelling of different scenarios (including different information flows and different data structures) based on different situations (conditions) within the same business transaction.
- To support modelling of different scenarios it is not sufficient to look only at the communication processes between the organizations. Rather it is necessary to analyse that internal processes of each organization to an extent that different information flows and different data structures can be determined. It must be clear under which circumstances which scenario has to happen. Therefore, the models must be able to support different views on operation calls and data structures.
- Information exchanged between organizations in EDI is often based on code sets. Models must allow the definition of code sets to be used in the EDI transaction. Furthermore, codes which are meaningful in a specific situation must be declared. Therefore, the models have to support relations between scenarios and instances of code sets.
- Phrasing conventions used in the models (e.g. class names, method signatures) have to be self explanatory to ensure a common understanding of the common business objects and, thus, the sharing of the models among organizations all over the world.

3 UML-based Guidelines to Support EDI Transactions

In order to support the requirements mentioned in the previous section, we follow the Unified Process to experience which parts of the Unified Process are meaningful to the modelling of EDI transactions (resulting in EDI specific guidelines) and to verify whether UML diagrams [2,15,18] are suitable to support the specification of BOV standards. In this section we present the resulting process for the business modelling workflow, the requirements workflow and the analysis & design workflow by means of a simplified 'Order from Catalog' example, which has been chosen from TMWG as demonstration project for OO-edi. Nevertheless it is important to note that the presented concepts are not yet agreed within TMWG.

3.1 The Business Modelling Workflow

The purpose of business modelling is to understand the structure and dynamics of the operations within a domain. It helps to ensure that all users, standards developers and further on software providers have a common understanding of the domain. In addition business modelling is used to derive the high level requirements needed to support the subsequent detailed analysis and eventual EDI solution. Note, that the business modelling workflow should allow insight into the business under consideration without any specific focus on EDI. It should generally describe what is performed in a business transaction. How EDI can support these business transactions will be part of subsequent workflows.

The business modelling workflow starts with a high level *definition of the vision and scope of the domain* to be considered (see Fig. 1). Furthermore, important terms used in the business should be covered in a *glossary* (e.g. the BuyerID: Seller assigned identification by which the seller uniquely recognizes a buyer). The vision and scope statement should allow to *derive the business actors* (roles of the organizations involved in the considered business transactions) *and the use cases* (main business transactions under consideration). Since the scope of the system is the inter-organizational communication between involved organizations, the use cases focus on communication processes between the actors and not on the internal operations performed by each actor (see Fig. 2).

Having found all use cases, the next step is to *detail each use case*. This covers a description of main activities performed in a use case and a high level description of information being exchanged. For example: To request a registration the buyer sends a registration request including his name and address, contact information and credit card information. This information could be used to design a first object model for each use case. We have omitted to do that, because in our project the business workflow

> The vision and scope of 'Order from Catalog' is described by five business transactions depicting the process of a Buyer executing a catalog order with a Seller. "Request Catalog" is an optional business transaction. A Seller may offer to provide to any potential Buyer an electronic version of the current Seller's catalog on request. "Register" depicts a first time Buyer initiating a relationship with a Seller by providing required buyer information, confirmed by receiving a Seller's Buyer ID from the Seller. "Request Price" (provide a price quote to the Buyer for selected product(s) on request) is an optional business transaction where the Seller may offer a price quote to a Buyer after a valid Seller's Buyer ID has been assigned. "Order Product" depicts the process of a Buyer ordering items from a catalog, having previously established a relationship with the Seller by providing Buyer information and receiving a Seller's Buyer ID (refer to "Register"). "Request Order Status" is an optional business transaction where the Seller provides order status information to the Buyer on request.

Fig. 1. Vision and Scope Statement for 'Order from Catlog'

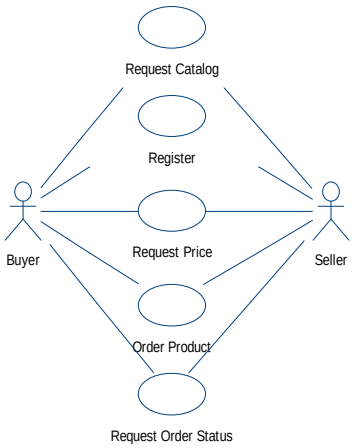

Fig. 2. Business Modelling Workflow: Use Case Diagram 'Order from Catalog'

(which is also optional in the Unified Process) should just allow a first insight into the business domain to understand what following workflows have to consider.

3.2 The Requirements Workflow

The objective of the requirements phase is for representative users and UN/CEFACT groups to come to an agreement on what an EDI solution for the selected domain should do. This stage normally takes a use case representing part of the business domain modelled in the business modelling workflow and refines the output for the area selected for the requirements modelling project. It concentrates on specifying requirements to a level that is good enough for users and standards developers to agree on what EDI solutions should provide. The process covers similar steps to those applied in the business modelling workflow. Whereas the business modelling workflow is independent of a specific communication technology, the requirements workflow concentrates on EDI specifics. Therefore, the steps are applied to a smaller area and a finer level of detail [19].

The extended team to provide the more detailed EDI requirements will include representatives who are knowledgeable about the business requirements of the domain as well as appropriate members working on UN/EDIFACT standards for the domain. At least one member of the team must be familiar with UML and this methodology.

The *vision and scope* of the business modelling workflow has to be refined to incorporate the EDI specific needs. It is essential to define an exact boundary of the system. It must be clear which business transactions (and which specific scenarios) of the business domain will be supported by EDI. Knowing the exact boundary it is necessary to define those actors who are inside the new boundary chosen for the requirements phase. Nevertheless, it is also essential to explicitly define those actors outside the boundary, but affected by inputs and/or outputs from processes within this boundary. Consequently, also information going in and coming out of the boundary have to be specified. Furthermore the business objects handled within the boundary must be identified. Since a deep insight necessary for these definition will often be gained when detailing a specific use case (which is a later step of the requirements workflow), the requirements workflow is considered as an iterative process.

Since EDI is the application-to-application exchange where no humans are involved

who can interpret certain semantics, there is a great sensitivity on the semantics of information exchanged. Hence, capturing a common vocabulary in a *glossary* is of great importance in the requirements workflow. Consider for example the term 'delivery date'. It seems that everyone might know what a delivery date is. But there is still a place for misinterpretations: Is it an exact, earliest, latest delivery date? Thus, a semantically complete definition must be stated in the glossary.

As mentioned earlier information exchanged in EDI is often coded. Consider the example of different delivery dates. In some situation it might be useful to state the exact semantics of a delivery date (exact, earliest, latest, ...) within an interchange. This can be realized with an instance of delivery data accompanied by a code specifying the kind of delivery date. Therefore, the glossary should also cover the definition of code sets including the various codes with a complete semantic definition. Another example for defining code sets would be the various codes needed to state the reasons for rejecting a certain request.

The next step of the requirements workflow is to *find the actors and use cases* (see Fig. 3) according to the boundary definition in the vision and scope statement [17,19]. Since EDI is the application-to-application information exchange, no users are involved in the inter-organizational transaction. Users might be involved in the operation of the internal system, which is not considered in the system in question. But input and output to the use cases is always sent/received by the information systems themselves. Consequently, the inter-organizational system has always to interface directly to the organizations' internal systems. To denote this fact, the use case model of the requirements model (which primarily focuses on EDI) does not depict actors, but the interfaces to the organisations' internal systems supporting the EDI transactions.

Taking a closer look on Fig. 3 it is easy to recognize that the definition of the use case 'Register Buyer' has been refined, because the use case takes advantage of another use case namely 'Verify Credit'. This is due to the fact that a seller wants to verify whether a buyer is credit-worthy or not. For this purpose, the seller contacts his bank to do this verification. Since this verification does not belong to the core processes of an order

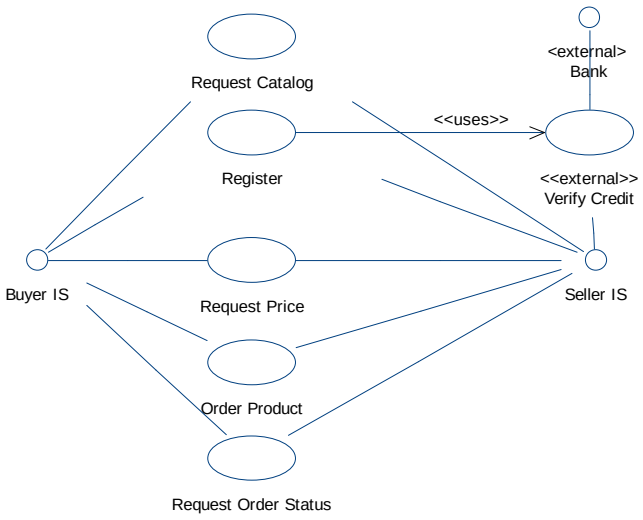

Fig. 3. Requirements Workflow: Use Case Diagram 'Order from Catalog'

from catalog it is outside the defined system boundary. Accordingly, the use case 'Verify Credit' must be defined in another system of EDI transactions. Therefore, the use case itself and the interface for the bank are stereotyped as 'external'. Nevertheless, it is necessary to analyse the required inputs and outputs from/to the external system.

The main function of the requirements workflow is to describe each use case in detail. We have developed a template for the purpose of a detailed use case description. Fig. 4 depicts the instantiated template for the use case 'Register Buyer'. The template has been designed to cover the following facts: For each use case the involved interfaces (actors) have to be defined. It must be clear which preconditions must be met before the use case can start and what initiates the start of the use case. Accordingly, one or more events must be specified which terminate the use case. The postconditions met by each of the end states have to be clarified.

Between the start event and the end event certain activities have to be fulfilled within the use case. Note that a use case can cover more than one scenario. This means that

Use Case Name:	Register Buyer
Summary:	In order to do further business with the Seller (obtain price quotes or order products), the Seller requires the Buyer to register and obtain a Buyer ID. Therefore, the Buyer provides the personal and credit information required for registration, and the Seller issues a Buyer ID.
Interfaces/Actors:	Buyer IS, Seller IS (internal) Bank (external)
Preconditions:	none
Begins When:	Buyer initiates the Registration process.
Description:	The Buyer initiates the registration process and documents the following information: Bill To details: Buyer name Bill to address (street, city, zip, country) Contact name (first, middle initial, last) Contact phone Ship To details (if different from Bill To info): Ship to address (street, city, zip, country) Ship to contact name (first, middle initial, last) Ship to contact phone Credit card info: Credit card number Credit CardHolder Name Credit Card Issuer Name Credit Card Type Credit Expiration Date Encrypted signature Respond-by date (date by which the Buyer wishes to receive the Buyer ID) The Buyer then sends this information to the Seller. When the Seller receives the request, the Seller checks the respond-by date. If the date has passed, the request is discarded. If the Respond-by date has not passed, the Seller validates the Buyers credit information (Uses Verify Credit Use Case). If the credit information is not valid, the Seller sends the Buyer a rejection notice containing the following information: Rejection reason code Rejection reason description If the Buyers credit information is valid, the Seller creates a Buyer ID for the Buyer. The Seller then sends a notice to the Buyer with the Buyer ID.
Ends when:	The Buyer receives a response from Seller, or the respond-by date is exceeded.
Exceptions:	none
Postconditions:	Buyer has a Buyer ID, a rejection of the Registration Request, or the request has been discarded

Fig. 4. Requirements Workflow: Use Case Description of 'Register Buyer'

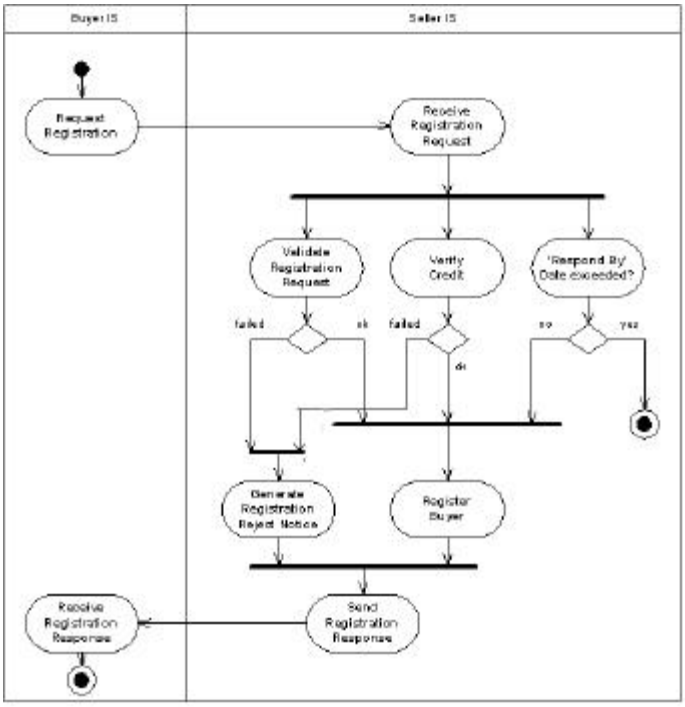

Fig. 5. Requirements Worksflow: Activity Diagram 'Register Buyer'

there might exist different paths through a use case (sometimes leading to different end states). The use case description has to capture all possible scenarios through a use case. To give a better understanding of the activities performed in a use case the textual description within the use case template is accompanied by an activity diagram for each use case (see Fig. 5). For each scenario the activities are given in the order they are regularly performed. It must be evident which conditions/decisions lead to different scenarios. Furthermore it must be clear which interfaces (actors) are involved in each activity. This can be defined by using swimlanes in activity diagrams (see Fig. 5) [15].

Finally, each use case description must cover a description of the business objects that are subject to the activities of the use case. The description in the use case template must allow to derive the business objects structure in a class diagram. Owing to space limitations we have omitted to depict a class diagram for the requirements workflow. However, class diagrams are especially focused in the following subsection.

The last step of the requirements workflow is to capture supplementary business specifications that are not already capturable in use cases. Such specifications include, for example, legal and regulatory requirements and application standards.

3.3 The Analysis and Design Workflow

The purpose of the analysis and design workflow is to transform the requirements identified in the requirements workflow into a design of the EDI system to-be [13]. The goal is to evolve a robust architecture of common EDI business objects. This architecture should allow organizations participating in EDI to build their internal EDI systems. Nevertheless, the design of the organizations' internal systems is not part of this analysis. But the architecture is designed to give advice on the business objects that the

internal systems have to support when participating in the considered EDI transaction. The workflow within each involved organization is analysed in a limited extent to capture different scenarios which have an impact on the overall EDI workflow. For example, if a request might result in different responses, the workflow at the responder's side will show the different situations that lead to different responses.

The team involved in the analysis and design workflow is made up of business modelling experts who have substantial know how of the focused business domain. They have to follow the described modelling conventions to ensure that the presentation of the architecture is consistent across the standardization bodies, independent software providers offering EDI software and EDI users.

The core part of the analysis and design workflow for defining EDI transactions is the use case analysis. For each use case identified in the requirements workflow a corresponding use-case realization is created in the design model. For the use case realization it is necessary to identify the classes that perform a use case's flow of events. The use case behaviour has to be distributed to those classes using use case realizations. Therefore, the responsibilities, attributes and associations of these classes must be identified [13].

The first task of the use case analysis is to supplement the descriptions of the use case to capture additional information needed to understand the behaviour of the system. In the requirements workflow the use case descriptions do not focus on the internal behaviour. Instead they only describe what the system is expected to do. Hence, this black-box description must be transformed in a white box description to define what the system does from an internal perspective. Due to space limitation we do not concentrate any further on this supplementary use case description for our demonstration example.

The next task is to identify a candidate set of analysis classes which will be capable of performing the behaviour described in the use case. Usually, there exist three types of analysis classes: entity classes, boundary classes and control classes. We take advantages of all three types in modelling EDI transactions. Entity classes represent the information exchanged in an EDI transaction. They are used to describe the structure of the 'virtual' business documents that are meaningful to the EDI transaction. Additionally, we find another type of entity classes in EDI transaction models. There must be entity classes covering information included in code sets. These classes comprise a fixed number of class objects. Each organization in the EDI transaction must be aware of these class objects, because the objects themselves are not interchanged, but the codes.

A boundary class intermediates the interface to something outside the system. Thus, the interfaces to the organizations' internal information systems are modelled as boundary classes. In addition to that, interfaces to external information systems that provide a service to the system are modelled as boundary classes. Control classes provide coordination behaviour in the system. In an EDI transaction model we use a control class for each role (which is usually carried out by one organization) participating in the transaction. Each control class is responsible for coordinating the EDI transaction from the viewpoint of the corresponding role. It is in charge of instantiating the entity classes and of interfacing the boundary classes.

In the class diagram of the demonstration example 'Register Buyer' (see Fig. 6) there are boundary classes for the interface to the buyer's information system, to the seller's information system and to the information system of the bank (used to validate the credibility of the buyer).

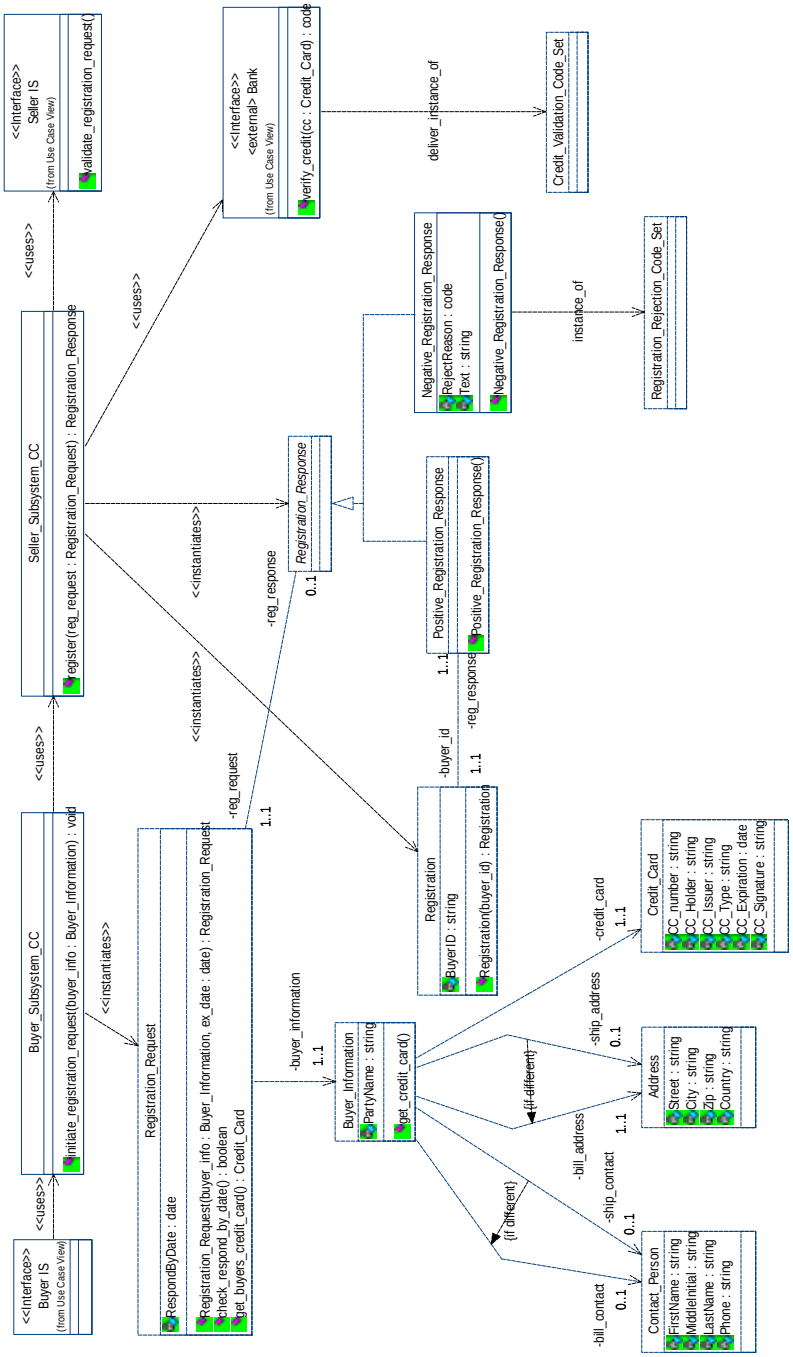

Fig. 6. Class Diagram for ’Register Buyer’

The entity classes present the structure of information exchanged in the registration transaction. Accordingly, the entity classes model a registration request to be sent from the buyer to the seller and a registration response to be sent back. For the registration response we use the concept of an abstract class. The concept of an abstract class is used in EDI transaction models to denote the fact that the structure of the information exchanged depends heavily on decisions made according to different situations. In our example the buyer expects a registration response to be sent back. But the information included in this response is dependent on whether the seller is willing to register the buyer or not. If so, the response will include the registration stating the buyer identification. Otherwise the response will capture the reason for the rejection. Thus, the concrete structure of the response is modelled in classes for positive response and negative response, which are subclasses of the abstract class for the general response. Additionally, the class diagram includes entity classes for the code sets for the registration rejection reasons and for the results of the credit verification.

Finally, we have a control class for the buyer subsystem and one for the seller subsystem. The control class for the buyer's subsystem is interfacing the buyer's information system and is responsible for creating the registration request. Similarly, the control class for the seller's subsystem is interfacing the seller's information system and the external bank and is in charge of creating the registration response. Furthermore, the buyer's control class uses that of the seller to set the registration request and to receive the response in return.

The next step of the use case analysis is to distribute the use case behaviour to the identified classes. The goal is to express the use case behaviour in terms of collaborat-

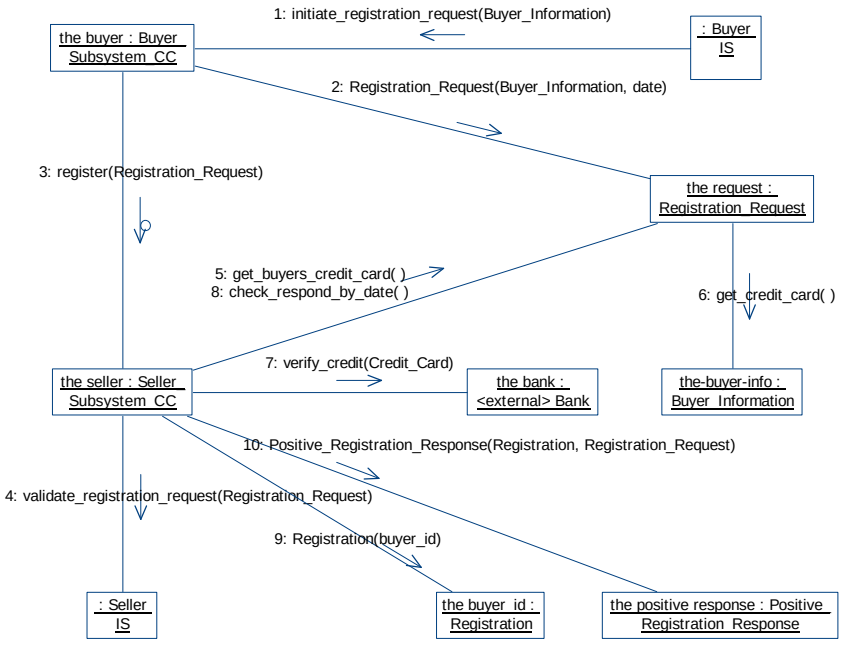

Fig. 7. Collaboration Diagram for a Successful Registration in 'Register Buyer'

ing analysis classes. For this purpose the responsibilities of each analysis class have to be defined. According to these responsibilities the actions performed by an object and the knowledge an object maintains and provides to other objects can be determined. To ensure that the responsibilities are correctly defined a collaboration diagram for each independent scenario must be created.

Fig. 7 shows a collaboration diagram for a successful registration in our demonstration example. The buyer's information system is willing to send a registration request. Therefore, it calls an instance of the control class of the buyer's EDI subsystem to create the request according to the stated buyer's information. The buyer's control object instantiates the entity objects of the registration request. Then it calls the register operation of the seller's control object. Note, that this operation call is a timeout function, because the buyer expects the response until a given response date. The seller's control object is now responsible for producing the registration response. Therefore, it calls the interface of its information system to validate the request. If successful, it determines the buyers credit card according to the registration request information structure. The credit card is input to the credit verification operation which is performed by the interface object of the corresponding bank. If credibility is given, the seller's control object will check whether the response date stated in the request is not expired. In this case the control object creates a registration object including the buyer's identification. This registration is then linked to the newly created positive registration response, which is the return value to the register operation called by the buyer's control object. Note, that the buyer as well as the seller will usually store the registration in their information systems. But we have not modelled these functions within the collaboration diagram, because they are not meaningful for the EDI transaction itself. These functions have to be considered in the design of an EDI system for a specific organization, which is out of scope for our purpose. Nevertheless, further collaboration diagrams have to be established for different scenarios leading to a negative response and for the scenario resulting from an expired response date.

The analysis of all the collaborations for the different scenarios has to lead to a consistent class diagram for the considered use case. The class diagram has to define all the attributes and operations assigned to the classes, the associations between the classes, as well as the existing dependencies. Fig. 6 depicts the final class diagram for the use case 'Register Buyer'.

In modelling EDI transactions we often take advantage of unidirectional associations to express the navigational direction for the exchanged information. For example, it is important to follow the link from the registration request to the buyer information to determine which buyer has sent the request. But for EDI transactions it is not meaningful to get all the registration requests a buyer has initiated, because in an EDI exchange only one registration request is sent.

Another important aspect of class definitions for EDI transactions covers the requirements for a value for certain attributes of an object. Unfortunately, it is not able to depict in the class diagram whether an object must have a value for a certain attribute or not. This information has to be included in the documentation of the class definition. Furthermore this documentation has to include rules for specifying the conditions under which an attribute value is optional. The same documentation must be applied for attributes resulting from existing associations. But in this case the cardinality already gives an advice on the optionality. Furthermore, it is essential to remark that all instantiated values for attributes of entity objects in EDI transactions should be frozen, because when instantiated they are sent to the partner organization that has to rely on this information.

The class diagram should also depict existing dependencies. This covers the 'uses' and 'instantiates' dependencies from/to control classes and boundary classes. But dependencies should also be defined in a class diagram whenever an attribute, a parameter or a return value refers to an entry of a code set. For example, the attribute reject reason of a negative response has to be a code mentioned in the registration rejection code set or the verify credit operation of the bank returns a code of the credit validation code set. Thus, these dependencies should be depicted in the class diagram. Unfortunately, our tool did not allow to assign the dependency exactly to the attribute and the operation, but only to the class.

The use case analysis has to be performed for each of the identified use cases leading to a class diagram for each use case. The final task of the analysis and design workflow for EDI transactions is to create an overall class design for the whole system. This means that a view integration of the different views resulting from different use cases has to be made. Firstly, this task has to handle relationships (include, extend, specialize) between the identified use cases. In addition to that, the reuse opportunities of classes in other use-cases have to be identified. For example, the control classes for the buyer and the seller will be used in each use case, but provide different services to each use case. Furthermore, generalization hierarchies between the analysis classes might be established. For example, each of the requests (registration, catalog, order, order status) might have some common attributes (respond by date) and methods (check respond by date), leading to a common superclass for requests. The design decisions made for the overall class design have finally be distributed to the use case specific diagrams.

The creation of BOV-related standards using UML is finished at this step. So far developed UML diagrams describe the business transactions to the greatest possible extent. Further design decisions can only be made on the basis of a concrete transfer mechanism, which is out of the BOV scope and part of the FSV layer. Nevertheless, the developed UML diagrams provide a consistent starting point for translation into a FSV related design.

4 Summary

In this paper we propose guidelines based on the Unified Process [13]and UML [15] to support the design of EDI transactions from a business-oriented viewpoint. According to the requirements of modelling EDI transactions using Unified Process and UML contributes to a consistent design of common EDI business objects in the following way: The business workflow helps in understanding the focused business domain. The requirements workflow describes the EDI specifics of the business domain. A glossary is used to ensure the semantically correct interpretation of EDI-specific and business terms. Furthermore, code sets for coded information used in an EDI interchange must be defined in the glossary. The vision and scope statement of the requirements workflow together with a use case diagram and supplementary use case definitions allow to exactly identify the boundaries of the EDI transaction. Within use case diagrams it is visible what is inside the boundary. Different stereotypes for interfaces allow to distinguish between interfaces to organizations' internal information systems (internal) and interfaces to external systems (external). External interfaces are often used when a service is provided by an external use case - also stereotyped in the use case diagram - defined in another EDI transaction. Nevertheless, a referencing mechanism to these external defined EDI transactions has not been developed by now. Furthermore, the requirements workflow uses activity diagrams to describe the main flow of events in a use case. Since activity diagrams use a rather simple notation to be easily understood by domain experts, they are able to state their requirements on the use case as well as to

validate the use case.

The analysis and design workflow finally is used to define the common EDI business objects. The entity classes in the class diagrams exactly define the data structure of the 'virtual' business documents being exchanged. Unfortunately, the requirement designators for attributes are not visible in the class diagram, but have to be documented in the class specification. The control classes controlling the EDI transaction for each participating organizations identify by their operation signatures what services the organization provides to the EDI transaction, what input is required to fulfil a service and what output can be expected. The 'instantiates'-dependency between control classes and entity classes defines which organization instantiates which 'virtual' documents or part thereof. Furthermore, the 'uses' dependencies show the interfaces among the control classes and between the control classes and the interfaces. Additionally, existing dependencies to code sets can be depicted in the class diagrams. The set of collaboration diagrams help to define different scenarios that might exist within a use case. The defined collaboration specifies the sequence of the activities in a use case and depicts different instantiations of entity classes based on different scenarios. The concept of abstract classes helps to model different data structures for the same type of return value. The scenarios leading to the concrete instantiation of the subclass of the abstract class are captured in the collaboration diagrams.

Consequently, the adapted Unified Process and UML provide meaningful concepts for modelling EDI transactions. But by now we have only gained experience from modelling one EDI transaction from scratch. Further investigations have to consider the reuse of patterns in various EDI transactions. Furthermore, concepts for linking these patterns and use cases assigned to different EDI transaction packages have to be developed. These facts become even more important, when a repository for capturing all EDI transactions for public reuse is established.

References

1. Berge, J.: The EDIFACT Standards. NCC Blackwell Limited, Oxford (1991)
2. Booch, G., Jacobson, I., Rumbaugh, J.: The Unified Modeling Language User Guide. Addison Wesley Object Technology Series, Reading (1998)
3. CEFACT: Report of the CEFACT Steering Group Chair to the CEFACT plenary. UN/Trade/CEFACT/1998/10 (1998)
4. D'Souza, D.F., Wills, A.C.: Objects, Components and Frameworks with Uml: The Catalysis Approach. Addison Wesley Object Technology Series, Reading (1998)
5. Eeles, P., Sims, O.: Building Business Objects. John Wiley and Sons, Inc., Chichester (1998)
6. Emmelhainz, M.A.: Electronic Data Interchange: A Total Management Guide. Van Nostrand Reinhold, New York (1990)
7. Fowler, M., Scott, K.: UML Distilled; Addison Wesley Object Technology Series, Reading (1997)
8. Hill, N.C., Ferguson, D.M.: Electronic Data Interchange: A Definition and Perspective. EDI Forum: The Journal of Electronic Data Interchange, Vol. 1, Issue 1 (1989) 5 - 12
9. Hruby, P.: Structuring Specification of Business Systems with UML. Proceedings of OOPSLA '98 - Business Object Workshop IV (1998)
10. Huemer, C.: Modeling Inter-Organizational Systems with UML. To appear in: the Proceedings of the 12th International Bled Electronic Commerce Conference, Bled, Slovenia (1999)
11. International Organization for Standardization: Open-edi Reference Model; ISO/IEC JTC 1/SC30 ISO Standard 14662 (1995)
12. Kande, M.M., Mazaher, S., Prnjat, O., Sacks, L., Wittig, M.: Applying UML to Design an

Inter-Domain Service Management Application. Proceedings of the International Workshop <<UML>>'98, Mulhause France (1998) 173 - 182
13. Kruchten, P.: Rational Unified Process. Addison Wesley Object Technology Series, Reading (1998)
14. McLeod, G.: Extending UML for Enterprise and Business Process Modeling. Proceedings of the International Workshop <<UML>>'98, Mulhause, France (1998) 195 - 204
15. OMG: UML Notation Guide, Version 1.1. OMG (1997) http://www.rational.com/uml
16. Raman, D.: Cyber Assisted Business - EDI as the Backbone of Electronic Commerce. EDI-TIE B.V., Hoofddorp, Netherlands (1996)
17. Rosenberg, D., Scott, K.: Use Case Driven Object Modeling with UML: A Practical Approach. Addison Wesley Object Technology Series, Reading (1999)
18. Rumbaugh, J: The Unified Modeling Language Reference Manual. Addison Wesley Object Technology Series, Reading (1998)
19. Schneider G., Winters J. P.: Applying Use Cases: A Practical Guide. Addisson Wesley Object Technology Series, Reading (1998)
20. TMWG: Reference Guide - 'The Next Generation of UN/EDIFACT'. CEFACT/TMWG/N010/R1 (1998)
21. TMWG: Assessment of UML for use as the Technique for Next Generation EDI Standards. CEFACT/TMWG/N026/R1 (1998)
22. TMWG: ASC X12/SITG Analysis of Modelling Techniques: IDEF 0, IDEF 1X and IDEF 3. CEFACT/TMWG/N034 (1998)
23. TMWG: Main characteristics of EXPRESS-G and assessment as a Technique for Next Generation EDI Standards. CEFACT/TMWG/N036 (1998)