**Global Networked Organizations**
Twelfth International Bled Electronic Commerce Conference
Bled, Slovenia, June 7 - 9, 1999

# Modeling Inter-Organizational Systems with UML

**Christian Huemer**

Institute of Applied Computer Science and Information Systems,
University of Vienna, Liebiggasse 4, 1010 Vienna, Austria
Tel. +43-1-4277-38443, Fax +43-1-4277-38449
ch@ifs.univie.ac.at

**Christian Huemer**

## Abstract

*In 1998 CEFACT decided to use modeling in the development process of EDI standards. On the basis of a careful analysis of various modeling techniques the Techniques and Methodologies Working Group (TMWG) decided that the Unified Modeling Language (UML) is be best suited for this purpose. The concepts of UML are appropriate to introduce modeling into the current UN/EDIFACT standard development process and also into a future process that may take advantage of new technology such as Object Oriented EDI (OO-edi). UML is called a modeling language, which uses a mainly graphical notation to express designs. It is not a method. Therefore, a process is needed to advice which steps to take in doing a design. In this paper we present a case study on modeling inter-organizational trade procedures based on an UML methodology. The design process utilizes the following UML notations: use case diagrams, class diagrams, activity diagrams and sequence diagrams. The methodology is introduced by means of the international trade transaction model.*

## 1. Introduction

In January 1998, the CEFACT Steering Group proposed the following resolution for adoption by the UN/EDIFACT Working Group (EWG): 'The UN/CEFACT Steering Group (CSG) resolves that business and information modeling is an essential requirement to the future of UN/EDIFACT and that the implementation of business and information modeling is a critical objective of the CEFACT strategy and its attendant work program' [4].

The Techniques and Methodology Working Group (TMWG) of UN/CEFACT investigated in three different modeling techniques - namely IDEF [18], EXPRESS-G [19] and UML [17] - as possible candidates in an EDI environment. According to a careful analysis of these techniques TMWG selected the Unified Modeling Language (UML) [13] as the technique for UN/CEFACT use in business process and information modeling [21].
UML is called a modeling language, not a method. Most methods consist of both a modeling language and a process. The modeling language is the mainly graphical notation that methods use to express designs. The process is their advice on what steps to take in doing a design [5, 6, 8]. What is required therefore, is a methodology for applying UML modeling techniques within the EDI standards development process. This methodology should be appropriate to be introduced into both the current UN/EDIFACT standards development process and within a future process that may take advantage of new technology such as Object Oriented EDI (OO-edi) [16]. Therefore, TMWG has selected the Rational Unified Process [8] as candidate process. Nevertheless, within the Rational Unified Process guidelines on how to exactly use the models must be specified.
TMWG takes the view that the requirements phase and the analysis phase of the current and the OO-edi standardization process are identical. 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 software [20]. Therefore, the first step is to provide modeling guidelines for the first two phases. Accordingly, the current focus is not on the development of software, but on using UML to describe business transactions on a conceptual level. Since UML has been designed to support the software development process, it was our goal to gain experience on the suitability of UML to support business modeling with a special focus on inter-organizational business modeling. Using UML for business modeling as well as for a following software process to design off-the-shelf EDI software would eliminate a paradigm shift necessary when using another design technique. Consequently, an UML-base methodology will ensure a consistent overall design process.

## 2. UML for Inter-Organizational Systems

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. UML can be used to ensure a better understanding of the inter-organizational system. UML models help to visualize an inter-organizational system and permit the specification of its structure and behavior. Furthermore, the models document the decisions that have been made [3].
In order to use UML for modeling inter-organizational systems we have to look at the features of UML. A premise of UML is that no single type of diagram can provide, on its own, a full representation of what goes on. So we need to use sets of related diagrams. Different types of diagrams represent different ways of approaching the problem, and see the system as being made up of a different set of 'things'. Each type of diagram is only capable of showing certain aspects of a situation - everything else is ignored [1].

Therefore, the first step is to select appropriate UML models to support the design process according to the requirements of inter-organizational business modeling. The following requirements have been identified:

- 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.
- The models must be able to capture the communication processes between the organizations. The order of the communication processes must be defined.
- The data structures supporting the information flows must be identified.
- The guidelines must support the modeling of different scenarios (including different information flows and different data structures) based on different situations (conditions) within the same business transaction.
- The sequence of the 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 modeling experience can deliver input to design their organizations' business practices and express their requirements on services from partners.
- The 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.

In order to support these requirements we apply use case diagrams, class diagrams, activity diagrams and sequence diagrams in our project. Furthermore, guidelines on how to use these UML diagrams to support the requirements must be developed. In this paper we propose guidelines that are well suited to support the design process of inter-organizational systems. It is important to note that these guidelines reflect the author's proposal to TMWG. But they are not agreed within TMWG.

In the following section we describe in detail the proposed guidelines for each of the above mentioned diagram types. Furthermore, we use step 3 of the international trade transaction model (ITT) *Preparation for Export* as practical example to illustrate the proposed guidelines. Note that we have not investigated the processes of the ITT itself, because it was our goal to show the applicability of UML to model the transactions and not to design the transactions from scratch. Thus, all the information about these transactions has been extracted from the UK's ITT model [15] and from the InterProcs ITT models [9], which use documentary petri nets to model processes [2, 10]. We follow the naming conventions of these models.

# 3. UML-based Guidelines

## *3.1 Use Case Diagrams*

A use case is a coherent unit of functionality provided by a system or class as manifested by sequences of messages exchanged among the system and one or more outside interactors (called actors) together with actions performed by the system. A use

case describes what a system (or a subsystem) does but it does not specify how it does it [3]. Nevertheless, result of value to an actor should be identified within a use case. An actor is a role of object or objects outside of a system that interacts directly with it as part of a coherent work unit (a use case). An actor element characterizes the role played by an outside object [11].
The use case model represents functionality of a system as manifested to external interactors with the system [12, 14]. In case of EDI the functionality regarded is the inter-organizational information exchange between organizations involved in a trade transaction. Therefore, the system to be developed must define the necessary interactions between organizations to perform trade transaction (which is the observable result of value).
Thus, only those services of an involved organization that provide interfaces to the overall inter-organizational system will be considered. Consequently, the internal operations performed by an involved organization are not of interest as long as they do not directly effect the inter-organizational transaction. Therefore, the organizations are in some respect outside the inter-organizational system and are modeled as actors according to their role in the international trade transaction. Accordingly, it is important to note that the boundaries of the inter-organizational system to be defined in the ITT use case model must be distinguished from the information system boundaries of applications used by the organizations.

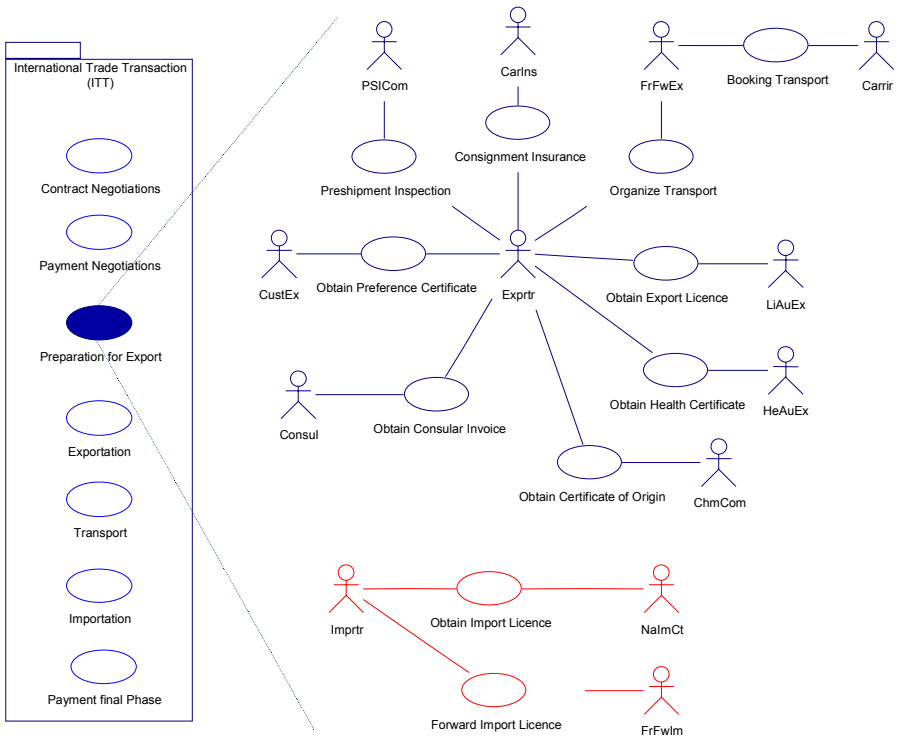


Figure 1: Use Case Model - Preparation for Export

For the ITT model we use several use case diagrams to show elements from the use case model. Each use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and the use cases [11]. The overall transaction comprises the following stages: Contract Negotiations, Payment Negotiations, Preparation for Export, Exportation, Transport, Importation and Payment Final Phase. The international trade transaction can be regarded as a package covering uses cases for each of these stages. This fact is depicted on the left side of Figure 1. Note that we have not depicted the actors involved in these use cases for reasons of simplicity. Each of these use cases will again be built by a set of subtasks resulting in use case models on the next level of abstraction. In Figure 1 we have chosen the task *Preparation for Export* to be refined in more detail. Therefore Figure 1 also shows the use case diagram for *Preparation for Export*. The actors have been assigned to the (sub-) use cases they are involved in.
Use Cases are an essential tool in requirement capture and in palling and controlling an iterative project. Capturing use cases is one of the primary tasks of the elaboration phase [6]. Therefore, we start with use case diagrams in order to define what is inside the scope of the business transaction and what is outside. But in order to capture all the requirements a solely graphical notation might not be enough. Therefore, a textual description covering the essentials of the use case should be added to each use case diagram. This text has to be clear enough for an outsider (business expert without modeling expertise) in order to be easily understood. For the purpose of modeling inter-organizational business transactions textual description covering a summary, involved actors, pre-conditions, start event, basic information flows, end event, post-conditions, exceptions and information about tracability seems to be appropriate. The following table provides an example description for the (sub-) use case *Consignment Insurance* that is a subtask of *Preparation for Export*.

| **Name** | **Consignment Insurance** |
|---|---|
| Summary | The exporter will insure the transport of consignments. An insurance contract will be established. |
| Actor(s) | The exporter, who requests insurance, and the carriage insurance. |
| Pre-Conditions | The Contract should be undertaken with an underwriter or insurance company of good repute and, failing express agreement to the contrary, be in accordance with the minimum cover of the INSTITUTE CARGO CLAUSES. |
| Begins When | The Use case begins when the exporter issues a request for insurance. |
| Description | The Carriage insurance draws up the contract<br>The contract is returned to the exporter for signature. |
| Ends When | The Use case ends when the exporter acknowledges the contract or when the contract is rejected by the exporter for any allowed reason. |
| Exceptions | Carriage insurance not able to insure the good/transport<br>trouble immediately found |
| Post-Conditions | The contract is approved to the exporter's satisfaction. |
| Traceability | Requirement {I.1,2,3,4,5,6} Requirement {II.100} |

## 3.2 Class Diagrams

A class diagram is a graphic view of the static structural model. It is a collection of (static) declarative model elements, such as classes, interfaces, and their relationships, connected as a graph to each other and to their contents. A *class* is the descriptor for a

set of objects with similar structure, behavior, and relationships. Classes have data structure and behavior and relationships to other elements. The attributes assigned to a class and the relationships to other classes define the data structure whereas the operations assigned to a class represent the common behavior of the class objects [11].
From a conceptual viewpoint two different sets of classes would be meaningful in describing the interactions in an inter-organizational system. The first set represents those classes which describe the data structures of information exchanged within each interaction. Consider for example the classes used to describe the data structure of an insurance contract, which is sent from an insurance company to the exporter in return to the request for consignment insurance. Although these classes are of great importance - especially for developing Open-edi standards leading to off-the-shelf software - we have not developed them at this stage of the project. This is due to the fact that we concentrate on the pure business processes in the first phase of the project.
The second set comprises those classes which interface the organizations' internal (information) systems and the inter-organizational system. Conceptually speaking these classes describe the services of the organizations' internal systems that are offered to the public world. Accordingly, there will be a class for each organization which provides a service in an international trade transaction.
Each interface class covers a description of the operations of an organization as offered to the public. These interface classes are part of the system, which is the inter-organizational system. Those operations that do not provide services to the public would not be part of the inter-organizational system and are therefore external to the inter-organizational system. Thus, operations must be assigned to classifiers outside the system. Since actors are classifiers that can perform operations, the organizations' internal operations should be assigned to the actors, who are external to the inter-organizational system.
From an UML purist viewpoint we need two different classifiers to describe organizations: interface classes covering operations that provide an external interface to the organization (and are internal to the inter-organizational system) and actors covering operations that are internal to the organization (and external to the inter-organizational system). Since we feel that a distinction between interface classes and actors is more confusing than helpful in the business analysis phase, we decided to use only one classifier concept for inter-organizational business process modeling.
Consequently, we have chosen the actor as classifier concept to model the behavior of an organization. Thus, we use the actor stereotype in the class diagram to denote all operations performed by an organization. This ensures an easier understanding for a non-UML expert. Furthermore, this solution strongly expresses the trade facilitation aspect of the UML models for ITT. This means that the ITT models will be valid even if the involved organizations do not make use of information systems. Otherwise, it will still be very easy to derive the requirements of the information systems to support the international trade transaction.
The differentiation between operations providing an interface to the inter-organizational system and those performing organization-internal tasks will still be possible even if only one classifier concept is used. The interface functions that standardize the organizations' external behavior are declared as public operations and can be invoked by an external organization. Organization-internal operations are always private to the respective actor. Private operations should be included for a better understanding of the business process.

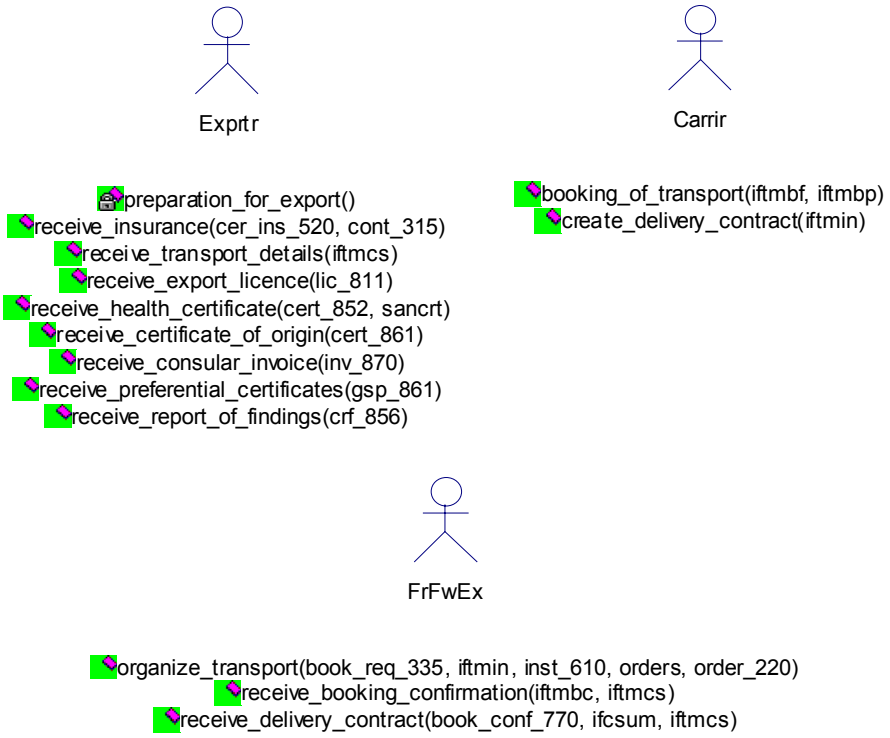


Figure 2: Class diagramms for interface classes

Figure 2 depicts three example definitions of organizations involved in the use case *Preparation for Export*: exporter (Exprtr), freight forwarder (FrFwEx) and carrier (Carrir). The definition of the exporter includes one private operation (characterized by the lock symbol in front of the operation's signature) that is called *preparation_for_export*. This operation is internal to the exporter and does not provide any service to other organizations. It implements all the necessary steps that an exporter has to perform to be prepared for export (see 3.3 Activity Diagrams). Its implementation also includes invocations of operations (services) offered by other organizations involved in this transaction. For example, to request transport it will call *organize_transport* of the freight forwarder. The organization of a transport is a service provided by the exporter and is consequently a public operation in the class diagram. To perform this task the exporter calls services from the carrier, which are the booking of the transport and the creation of a delivery contract.
There is a public operation for each main business function an organization contributes to the overall transaction in question. In addition, public operations that do not have a corresponding major business function are assigned to the actor. These public operations reflect the asynchronous way of business interactions. For example, if an exporter asks the freight forwarder to organize the transport, he is expecting an answer from the freight forwarder. The exporter must be prepared for receiving all answers to service requests. Thus, there will be public operations to receive returns to all service

requests. If the freight forwarder has completed the organization of the transport, he will send the details of the transport to the exporter by invoking receive_transport details, a public operation assigned to the exporter (see 3.3 Activity diagrams).
As mentioned above we have concentrated so far on the business processes in the international trade transaction. Nevertheless, it is important to note that each interaction results in an information exchange. The information expected by an actor to fulfill his services must be defined for each interaction. Thus, the input for an operation is declared within the parentheses of its signature. Since we have modeled the transaction in an asynchronous way an operation has no output, instead the output is provided as input to the invocation of a receiving operation of the originator. The way the input to operations is described depends on the purpose of the transaction model.
At this point of our project we just look at the business processes to be carried out by organizations and do not concentrate on the data structures which support these processes. It is our opinion that a careful analysis of the business processes has always to precede the definition of the data structures. Having carefully analyzed the business processes in a succeeding step the data structures to support these processes have to be identified. This can be argued by the fact that the data structures have to support the business processes and not the business processes have to support the data structures. Since our project focuses on business processes at a conceptual level the information exchange can be described at a very high level of abstraction, e.g. by listing the documents to be exchanged. Therefore UML models might be used to define the business processes even if no interfaces to information systems are considered and the information exchange definition references only paper documents. In our UML model we assume that organizations make use of information systems, but we still concentrate on business processes. This means that we do not care about the data structure of information to be exchanged, but reference to existing EDI messages. Therefore, input to operations is defined by already existing EDI messages. The EDI messages which are meaningful for each operation have not been validated by ourselves, but have been extracted from the documentary petri nets approach used in the InterProcs project [9].
In a further stage of the project we will also concentrate on the data structures of information to be exchanged. The data structure will be based on UML class diagrams to define data items and their interrelationship without any concern about the physical representation in an information exchange. This approach corresponds to the business-oriented view (BOV) of the Open-edi reference model [7]. Having defined the conceptual structure of data to be exchanged, it would be easy to map the data structure to a physical design required by an electronic data interchange method. Consequently, the UML models can serve as basis for various purposes: development and maintenance of EDI messages and implementation guidelines, definition of XML/EDI messages or development of component-based architectures to support EDI.
Consequently, the purpose of the so far developed class diagrams is to give an overview of the services provided by each organization. Note that the definition of the class diagrams does not precede the definition of activity diagrams and sequence diagrams. Rather the definition of these diagrams has to occur in parallel.

## *3.3 Activity diagrams*

An activity diagram is a special case of a state diagram in which states are action states and in which transitions are triggered by completion the actions in the source states. The

entire activity diagram is attached to a class or to the implementation of an operation or a use case. The purpose of this diagram is to focus on flows driven by internal processing [11].

In the proposed guidelines we use activity diagrams to model activities to take place within both public and private operations. Consequently, each activity diagram focuses on the implementation of an operation defined in the class diagram. An activity diagram defines all the activities and their sequence to perform an operation. But activities are not necessarily in a sequential list. Decision activities are used to model alternative flows within one operation. Synchronization bars denote that several activities can occur in parallel.

Since each activity diagram implements an operation that belongs to exactly one actor, it is internal to that actor. In a business context this means that activity diagrams outline actions to be performed internally to one organization. As a consequence, they can be built from the viewpoint of the organization in question. This fact might help when designing an inter-organizational system, because activity diagrams assist to detect the business and the communication requirements of a certain organization in fulfilling its task.

To perform a certain activity it is often necessary to demand a service of another organization. A service provided by another organization is defined as an operation assigned to the actor of the relevant organization. Therefore, the activity requiring a service and the operation accomplishing this service must be synchronized. This synchronization is realized by indicating the invocation of a foreign operation (*^Actor.operation()*) beside the activity in the activity diagram. But we have not indicated the input to the operation, which is defined in the corresponding class diagram. Note, that for an organization it is not of interest how the co-operating organization performs its service. The only relevant information is that the co-operating organization offers the required service and the input the co-operating organization expects to provide the service.

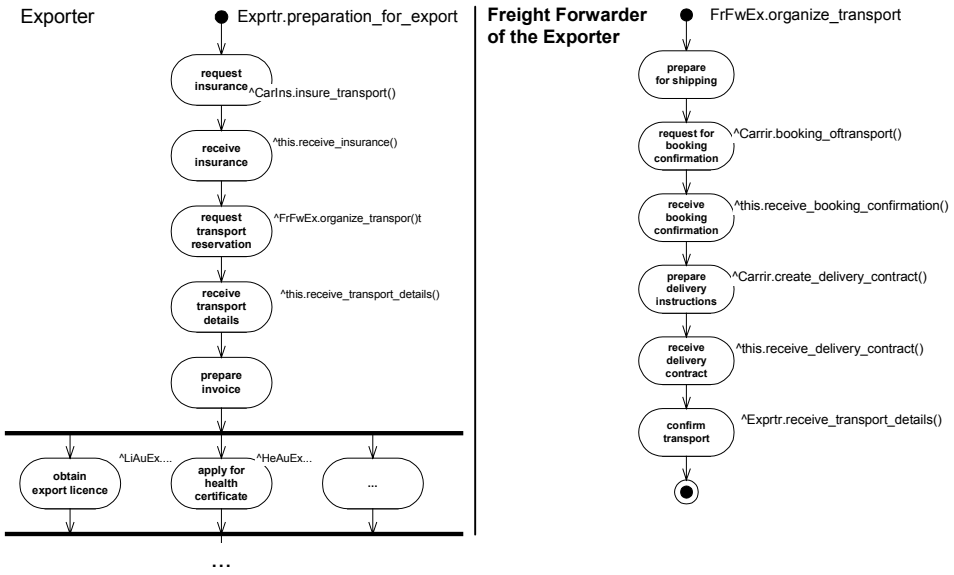


Figure 3: Examples of Activity Diagrams

Furthermore, a request for a service will most likely result in an answer to that request. Hence, the requesting activity is usually followed by an activity expecting the response. This activity usually corresponds to a public operation assigned to the own actor, which is invoked by the implementation of the co-operating organization's operation. This fact is denoted in the activity diagram by indicating the own operation (*^this.operation()*) beside the activity. Note that we use the keyword *this* instead of *actor* to announce that it is the own instance of the actor and not another instance of the same actor definition (e.g. our export organization and not another exporter).
The whole inter-organizational system gains its cohesion from well-designed operation calls among the involved actors (see also 3.4 Sequence Diagrams). Thus, there is a need for harmonizing the originator's expectations on a service including the deliverable input and the responder's readiness to fulfil the service including the expected input. The required services can be derived from the originator's activity diagram(s) and the offered services are defined in the responder's class diagram. Therefore, the development of an inter-organizational system requires an iterative design process between activity diagrams and class diagrams.
Figure 3 depicts two examples of activity diagrams. On the left an extract of the activities to be executed by an exporter when preparing for export (*Exprtr.preparation_for_export*) is shown. The organization of the transport by its freight forwarder (*FrFwEx.organize_transport*) is given on the right.
The exporter starts his internal operation to prepare for export by requesting a carriage insurance company to insure the consignment. For this purpose it calls *CarIns.insure_transport()*. After having received the insurance license via its own operation *this.receive_insurance()*, the exporter requests its freight forwarder to make a transport reservation by invoking *FrFwEx.organize_transport()*. The freight forwarder organizes the transport according to the description below. The exporter expects to receive the details of the transport (*this.receive_transport_details()*). Afterwards the exporter prepares the invoice and will now be able to go on with further activities that might be executed in parallel. Due to space limitations we omitted the remaining activities.
When the exporter calls the operation *organize_transport()* the freight forwarder will execute the activities as defined in the underlying activity diagram. First of all he prepares himself for shipping. Then the freight forwarder requests the carrier to book the transport (*Carrir.booking_of_transport()*) and waits for the booking confirmation (*this.receive_booking_confirmation()*). Afterwards the freight forwarder prepares the delivery instructions including a call to the carrier to create a delivery contract (*Carrir.create_delivery_contract()*). After the freight forwarder receives the delivery contract (*^this.receive_delivery_contract()*) he has finishes most of its activities. Finally, the freight forwarder confirms the transport and sends the transport details back to the exporter by invoking *Exprtr.receive_transport_details()* of the exporter.

## *3.4 Sequence Diagrams*

A sequence diagram represents an interaction, which is a set of messages exchanged among objects within a collaboration to effect a desired operation or result. It shows the objects participating in the interaction by their “lifelines” and the messages that they exchange arranged in time sequence. A sequence diagram has two dimensions: the vertical dimension represents time, the horizontal dimension represents different

objects. Sequence diagrams come in several slightly different formats intended for different purposes [11].
In the proposed guidelines we use sequence diagrams to give an overall picture of the communication between organizations involved in a transaction defined by a use case. As mentioned in the previous section the whole system gains its cohesion from well-designed operation calls among the involved actors. Activity diagrams describe the necessary interactions only from the viewpoint of a single organization. By representing the involved actors on the horizontal dimension of a sequence diagram it is possible to show all operation calls among the actors in order to carry out an inter-organizational trade transaction.
Since we concentrate on the invocation of operations, we do not include any internal activities of the actors (e.g. prepare invoice that is internal to the exporter). The requirement for sequence diagrams to present the timeline on the vertical dimension must in some respect be undermined. The fact that an actor might call multiple operations in parallel (no explicit order to call the services is defined) cannot be represented in sequence diagrams. Furthermore, it is rather complex to represent alternative paths in the implementation of an operation resulting in different operation calls. In this case multiple sequence diagrams must be developed.
For the above mentioned reasons the complete information on invocation of operations and their interdependencies is only described in the full set of activity diagrams. Sequence diagrams do not add any further information to the definitions made in the class diagrams and in the activity diagrams. Nevertheless, sequence diagrams provide a meaningful concept to aggregate the information included in the full set of activity diagrams and in the class diagrams in order to show the interactions between multiple actors.
Figure 4 presents an extract of the sequence diagram for the use case *Preparation for Export*. It shows the interactions between exporter, carriage insurance, freight forwarder and carrier. For a detailed description of these interactions we refer to the section 3.3 Activity Diagrams. For this example we have omitted to show any further interactions between the exporter and other organizations.

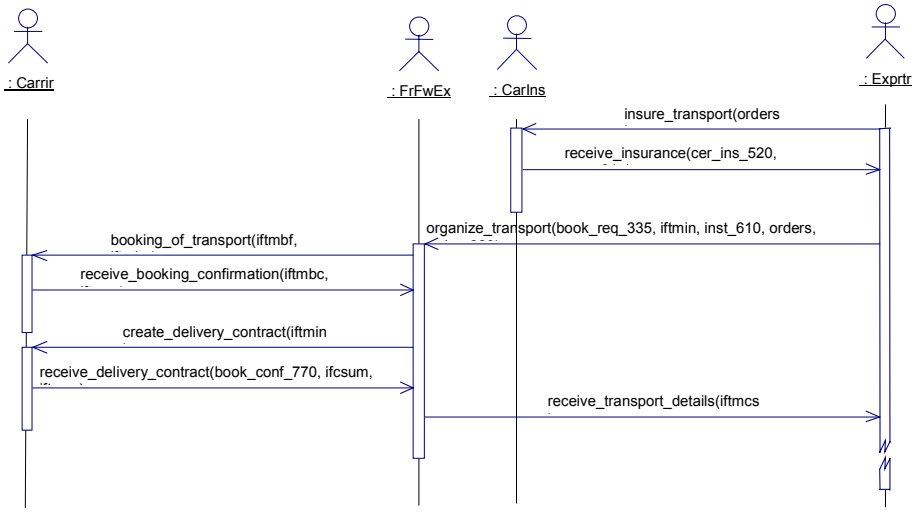


Figure 4: Example of a Sequence Diagram

# 4. Summary

In this paper we propose guidelines based on UML to support the design process of inter-organizational systems. The approach takes advantage *of use case diagrams*, *class diagrams*, *activity diagrams* and *sequence diagrams*. The experience on the applicability of using UML for the design of inter-organizational systems can be summarized as follows:

The boundaries of the inter-organizational system can be defined by use case diagrams and an accompanying structured description including pre-conditions, start event, basic information flows, end event, post-conditions, exceptions and information about tracability. The main activities (and their sequence) that have to be carried out by an organization to fulfill a service can be modeled with activity diagrams. A separate activity diagram for each main business function allows to describe the inter-organizational system from the viewpoint of each organization. The contribution of each organization to the whole process is defined by all the activity diagrams for its services. Furthermore, activity diagrams use a rather simple notation to be easily understood by domain experts in order to state their requirements on the process as well as to validate the process. A total overview of the services provided by each organization and the expected input to these services is given in the class definition of the organization in question.

The communication processes between the organizations are captured on the one hand by the above mentioned activity diagrams and on the other hand by sequence diagrams. To incorporate the communication processes into activity diagrams we extend their notation by mentioning the required service of another organization beside the activity in question. This defines the communication processes always from the viewpoint of one organization. Sequence diagrams that depict all the service calls between all organizations give an overall picture of the whole process. A weakness of sequence diagrams is the fact that it is not easy to denote operation calls that can occur in parallel. Furthermore, sequence diagrams can get overloaded and confusing when alternative paths to the information flow are described. In the current project we only have modeled a 'perfect' scenario without alternative paths. But when extending the project to real world situations further investigations on how to incorporate this information into sequence diagrams must follow.

Furthermore, the current models have to be extended to incorporate the data structures supporting the information flows. UML allows to define these data structures within class diagrams. Nevertheless, it will be a challenge to describe views on the data structures for different scenarios in a clearly arranged manner.

It should be noted that use case diagrams, class diagrams, activity diagrams and sequence diagrams were able to provide the concepts according to the requirements of our current state of the project. When incorporating data structures into the approach or going from the conceptual level to the implementation level, it has to be verified if other UML concepts and diagram types must be supported by the guidelines as well.

## References


1. Benson T. (1998). Short Strategic Sudy: Enabling Technologies - UML (Final Report). CEN TC 251 Health Informatics, CEN/TC 251/N98-082
2. Bons R.W.H., Lee R.M., Wagenaar R.W. (1995). Modelling Inter-organisational Trade Procedures Using Documentary Petri Nets. Proceedings of Hawaii International Conference on System Sciences (HICSS) 28, Hawaii, January 1995
3. Booch G., Jacobson I., Rumbaugh J (1998). The Unified Modeling Language User Guide. Addison Wesley Longman
4. CEFACT (1998). Report of the CEFACT Steering Group Chair to the CEFACT Plenary. Trade/CEFACT/1998/10
5. D'Souza D.F., Wills A.C. (1998). Objects, Components and Frameworks with Uml: The Catalysis Approach. Addison-Wesley Object Technology Series
6. Fowler M., Scott K. (1997). UML Distilled. Addisson Wesley
7. ISO (1996). The Open-edi Reference Model. IS 14662, ISO/IEC JTC1/SC30
8. Kruchten P. (1998). Rational Unified Process. Addison-Wesley Object Technology Series
9. Lee R.M., et al. (1998). WWW-Homepage of InterProcs. http://abduction.euridis.fbk.eur.nl/projects/InterProcs.html
10. Lee R.M. (1998). Distributed Electronic Trade Scenarios: Representation, Design, Prototyping. Internal Report RP 1998.09.01 of Erasmus University Research Institute EURIDIS, Rotterdam, The Netherlands
11. OMG (1997). UML Notation Guide, Version 1.1. OMG http://www.rational.com/uml
12. Rosenberg D., Scott K. (1999). Use Case Driven Object Modeling with UML: A Practical Approach. Addison-Wesley Object Technology Series
13. Rumbaugh J. (1998). The Unified Modeling Language Reference Manual. Addisson Wesley Object Technology Series
14. Schneider G., Winters J.P., Jacobson I. (1998) Applying Use Cases: A Practical Guide. Addisson Wesley Object Technology Series
15. SITPRO. The International Trade Transaction Model (ITT Model). http://www.unece.org/trade/itt/uk/intro.html
16. TMWG (1998). Reference Guide - "The Next Generation of UN/EDIFACT" (Revision 12). CEFACT/TMWG/N010/R1 http://www.harbinger.com/resource/klaus/tmwg/documentlist.html
17. TMWG (1998). Assessment of UML for use as the Technique for Next Generation EDI Standards. CEFACT/TMWG/N026/R1
18. TMWG (1998). ASC X12/SITG Analysis of Modelling Techniques: IDEF0, IDEF1X and IDEF3. CEFACT/TMWG/N034
19. TMWG (1998). Main characteristics of EXPRESS-G and assessment as a Technique for Next Generation EDI Standards. CEFACT/TMWG/N036
20. TMWG (1998). Business and Information Modelling Impact Study. CEFACT/TMWG/N043/R1
21. TMWG (1998). Explanation of TMWG decision to use UML for Business Process and Information Modeling. CEFACT/TMWG/N65