Modeling Business Collaborations in Context

Birgit Hofreiter & Christian Huemer
Department of Computer Science and Business Informatics
University of Vienna, Liebiggasse 4, 1010 Vienna, Austria
{birgit.hofreiter,christian.huemer}@univie.ac.at

Abstract. Standard e-business document types are usually too ambiguous due to an overwhelming choice of data elements. Business partners must agree on a shared subset and adapt their application interfaces accordingly. Small and medium enterprises (SMEs) cannot perform this task. UN/CEFACT's Modeling Methodology (UMM) provides a methodology to define unambiguous business collaborations allowing software vendors to integrate corresponding APIs into their business software. Business collaborations depend on their business context, i.e. parameters describing the business environment. Instead of developing different models for each specific business environment, we need a single model for a specific business goal clearly specifying the context variations. In this paper we extend UMM to show how a generic business collaboration is adapted to different business contexts. This is demonstrated by a case study on ordering/selling books as well as tourism products.

1 Introduction

Business-to-business e-Commerce (B2B) is not particularly new. For a long time the field was dominated by electronic data interchange (EDI) standards [6]. However, small and medium enterprises (SMEs) failed to implement EDI due to the complexity of EDI standards, as UN/EDIFACT or X12. XML seemed to be the solution to traditional EDI problems. However, the most important limitation still exists [9]: Standard document types are overloaded. Usually, a specific partnership requires about 3% of the data elements of a document type. Thus, business partners have to agree on a corresponding subset and additional rules. Each partner must implement a mapping between the in-house system and the exchange format that follows the agreement.

This paper is based on the following premises: SMEs will never be able to implement this type of mapping. Consequently, SMEs need software packages that provide both the business functions needed (e.g. purchase order handling) and the B2B functionality (e.g. ordering goods from a business partner). Only if these functions go hand in hand together in low cost commercial off-the-shelf software (COTS), we will see B2B e-commerce to take off.This requires a paradigm shift for producing B2B standard specifications. These specifications must unambiguously describe business processes among two or more partners, which we call business collaborations. Only unambiguous definitions allow software vendors to integrate these collaborations into their software products. Integrating a set of APIs requires both unambiguous data structures and an unambiguous choreography. The choreography defines the dynamic aspects of the collaboration in terms of agreed information flows, whereby each exchange leads to an agreed business state change.

A methodology similar to a software process is needed to develop unambiguous business collaborations. For this purpose, the United Nation's Centre for Trade Facilitation and e-Business (UN/CEFACT) has developed UN/CEFACT's Modeling Methodology (UMM) [15,8], which is based on the Unified Modeling Language (UML) [1]. It describes a method and supporting components to capture business process knowledge, independent of the underlying implemented technology so that the business acumen is retained and usable over generations of implemented technology.

Having been involved in the development of UMM since day one, we present its basic concepts and strengths. However, UMM is an ongoing effort that still needs improvements. Its current version 12 perfectly guides through the process in case of a very specific business goal in a detailed business environment. This would lead to a multitude of models. Instead of developing different models for each specific business environment, we need a single model for a specific business goal clearly specifying the context variations of the business environments. Version 12 does not specify a methodology for reusing business objects to assemble the information exchanged. Furthermore, we are still missing an example which spans all over the UMM process steps. Hence, this paper adds the following aspects to the current version of UMM:

- Binding context drivers to each model. Context drivers represent a set of parameters to describe the business environment. This concept ensures the specification of generic business models with rules to adapt to specific business environments.
- Assembly of information exchanged based on ebXML core components.
- A case study for a simple punch-out catalog that spans over all UMM steps and adapts to two different business environments.

The remainder of this paper is structured as follows: Section 2 presents an overview of other approaches considering business processes in a B2B environment. Furthermore, it motivates the use of an object-oriented approach and UMM in special. An introduction into the process steps of UMM is presented in Section 3. The case study of a punch-out catalog for book and tourism products in Section 4 follows these steps. A summary in Section 5 concludes the paper.

2 Related Work

The idea of standard business scenarios and the necessary services to support them was first created by the Open-edi reference model that became an ISO standard in 1997 [10]. Thereby Open-edi separates the "what" in the Business Operational View (BOV) from the "how" in the Functional Service View (FSV). The BOV covers the business aspects such as business information, business conventions, agreements and rules among organizations. The FSV deals with information technology aspects supporting the execution of business transactions.

Since Open-edi is not an implementation specification, projects started to deliver "Open-edi"-compliant implementations. The InterProcs system [11] is a prototyping environment to support the design and execution of Open-edi scenarios. The language used to represent these trade scenarios is based on petri nets. Petri nets provide a well known formalism to model and execute workflows. Since B2B always involves a type of workflow between organizations, petri nets are appropriate to model these interorganizational workflows. This concept is also used by other authors (cf. [18,19,13,12]). Although petri nets are well suited for modeling an unambiguous choreography for business collaborations, they do not model the structure of the information exchanged.

The object-oriented approach cares about both the statics and dynamics of a system. UN/CEFACT's Techniques and Methodologies Group (TMG) is developing an object-oriented methodology to define the BOV layer of Open-edi. The work started off in 1998, but the UMM spec is considered as a living document with multiple revisions. UN/CEFACT's, which is known for their UN/EDIFACT standards, builds their next generation of EDI on top of UMM. When UN/CEFACT and OASIS started the ebXML inititative, it was UN/CEFACT's vision UMM is used to create BOV standards and XML is used as key concept in the FSV layer. Accordingly, UMM is ebXML's modeling methodology, but it is not a mandatory part of ebXML (cf. [4]). Especially,

during the 18-month initiative of ebXML, UMM made a lot of progress. TMG and ebXML members, such as SWIFT, TM Forum, EAN*UCC, and RosettaNet participated in the development. In 2000 the copyrights of the Business Collaboraton Framework (BCF), used by RosettaNet were transfered to UN/CEFACT and the BCF was merged into UMM's version 10. The ebXML core components specification includes the concept of customizing generic components to the special needs of a business context. This concept does not exist in UMM version 12. In the following sections we describe how to merge this concept into UMM.

Since UMM stops at the BOV layer, a transformation to an IT solution on the FSV layer is required (c.f. [2]). Today the IT solution of choice is definitely web services. The web service community with their work on web service choreography represents another group focusing on business processes in B2B. However, their conceptual work is strongly intermingled with the XML representation languages (e.g. BPEL, WSFL, WSCL, WSCI) that are of their primary interest [20]. Currently, there only exists a project proposal for a conceptual model for developing and describing web services and their composition [5]. Since UMM is independent of the underlying technology, it is as candidate to fill this gap. For this purpose mappings between UMM and the various web service languages must be developed, which is out of scope of this paper.

3 The UMM Process

UMM is a UML-based methodology for describing Open-edi scenarios. It concentrates on the business semantics of the BOV of Open-edi. The steps in UMM are similar to a software development process. Since it does not (yet) focus on the FSV of Open-edi, steps from the implementation onwards are not considered. Hence, UMM provides the business logic to develop middleware for B2B as well as software components to interface the B2B processes. In the following subsections we describe the steps of UMM. Each step is based on a meta model, i.e. a UML profile for UMM defining a set of stereotypes with tagged values and constraints [16]. We recommend to read this paper together with the UMM profile mentioned above.

3.1 Business Domain View

The first workflow of UMM is used to gather existing knowledge. It identifies the business processes in the domain of the business problems that are important to stakeholders. It is important at this stage that business processes are not constructed, but discovered. Stakeholders might describe intra-organizational as well as inter-organizational business processes. Both are valid and recorded. However, the description concentrates on so-called business interface tasks, where a business communicates with its partners. All the discovered business processes are classified according to a predefined classification schema. The final result of the business domain view allows a business process analyst to find opportunities for business collaborations that are constructed in the following workflows.

3.2 Business Requirements View

The goal of the business requirements view is to identify possible business collaborations in the considered domain and to detail the requirements of these collaborations. Business collaborations span over multiple business processes discovered in the previous workflow. Thus, a use case for a business collaboration must consider the views of different stakeholders. The description of the use case must present an harmonized view on the business collaboration being developed.

The business goals as described in the business collaboration use case description are realized by a business collaboration. In the current version of the UMM meta model

there exists a 1:1 relationship between a business collaboration use case and its realization in the business collaboration. We suggest a 1:n relationship: A business collaboration use case will describe a generic business goal independent of the business environment. The realizations in the business collaborations are dependent of the business context of its environment. It seems to be straight forward that a generic business collaboration is realized in different business contexts (see Fig. 2 of the example). The business environment for each business collaboration must be documented. It is described by a set of tagged values representing context drivers. Candidates for these context drivers were identified during the work on ebXML core components [17]: business collaboration, business transaction, product classification, industry classification, geopolitical context, official constraint, business process role, supporting role and system capabilities.

3.3 Business Transaction View

The Business Transaction View represents the view of the business process analyst. In the first step the business collaboration is modeled according to the corresponding use case description. For each identified business collaboration a corresponding choreography, the so-called business collaboration protocol is modeled by an activity diagram (see Fig. 3). A business collaboration protocol designs one or more business transaction activities. A business collaboration protocol is not a transaction and is used in cases where a transaction rollback is inappropriate. For each business transaction the maximum performance time is documented. If this time is exceeded, the initiating partner has to send a failure notice. Furthermore, for each business transaction activity it is defined whether or not more than one business transaction can be open at one time.
Each business collaboration protocol defines a choreography in a given business context. Thus, we recommend to add context drivers to the business collaboration protocol as tagged values. A business collaboration protocol might be generic, i.e. valid in multiple business contexts. However, some of its business transaction activities and transitions might be valid in a limited context. In this case context drivers are assigned to the corresponding business transaction activities or transitions (see Fig. 5 of the example).

The next step in the Business Transaction View is to detail each business transaction activity by a separate activity graph called a business transaction (see Fig. 6 of the example). Again the business environment is added to the business transaction as tagged values. The specification of business transactions builds up on the experience gained from the development of RosettaNet Partner Interface Processes (PIPs). A business transaction is made up of a requesting business activity performed by the initiating partner and a responding business activity performed by the responding business partner. The requesting business activity outputs a business document (represented by a object flow state) that is input to the responding business activity. A business document created by the responding business activity and returned to the initiating business activity is optional. A business transaction follows one out of six different types of patterns.

By analyzing the business transaction it is important to define the following values for both requesting and responding activities: Time to Acknowledge Receipt, Time to Acknowledge Acceptance, Time to Perform, Authorization Required, and Non-Repudiation of Origin and Content. The values for Non-Repudiation of Receipt and for Recurrence are defined only for the requesting business activity. Note, acknowledge of receipt is sent after grammar validation, sequence validation, and schema validation. Acknowledge of acceptance is sent after an additional content validation. Recurrence is the number of retries in case of control failures. The different types of business transaction patterns also differ in the default values for the parameters mentioned above.

Each business document exchanged in the business transactions is modeled in a class diagram. The following flags are set for each class and/or attribute: is confidential, is tamper proof and is authenticated. In order to guarantee reusability, the business documents must be built by common business objects. Unfortunately, the current version of UMM does not reflect this requirement. The meta model only defines that the business information exchanged is built by recursively structured information entities. We recommend that these business objects are built on the basis of ebXML core components [17]. In this specification a core component is defined as "*a building block that contains pieces of business information that belong to a single concept. Core components are characterized by the fact that they appear in many different circumstances of business information and in many different areas of business.*" Aggregate core components become business objects and basic core components their attributes. The type of an attribute corresponds to the type of the core component, which covers a content component and complementary components. When using a core component in an interchange, one must set it into the right business context - it becomes a so-called business information entity. In other words the core component is customized according to the business environment. Similarly, instead of simply reusing ambiguous business objects, an unambiguous customization is defined by setting the context drivers. Thus, software providers will not face an overwhelming choice of components, but must support the customizations for the business environment of their software packages. The detailed technique to model business documents is best understood in the case study (cf. Table 2 and Fig. 7).

3.4 Business Service View

The fundamental principle of the business service view is to describe the business collaborations between network components. The business service workflow does not add any new information. Accordingly, the business service view artifacts are automatically created from the information gained in the previous workflows. For each role appearing in any business transaction a network component is created. In an application-to-application environment these components communicate with each other. Each business transaction is mapped to a service transaction defining the exchanges between the components (see Fig. 8). Exchanges are either business document exchanges or business signals sent as acknowledgment or failure notices.

Note that each business activity of a business transaction will result in an operation assigned to the corresponding network component. A service collaboration spans over all service transactions. A network component for a given role has to support all the operations defined in any business transaction (see Fig. 9 of the example).If a company is capable of a role in a business collaboration, it has to support all the operations of the corresponding network component and must keep the choreography defined in the collaboration. A company might support one business context, but not another. It follows that each network component must define the context drivers it supports in its profile.

4 The UMM Case Study

In this Section we present the UMM by means of two case studies. The first one is the order management of books and the second one is the order management of tourism products like hotels, flights etc. Due to space limitations we cannot describe the examples in all its real world requirements. Therefore we have chosen a limited functionality that still allows to understand the concepts of UMM. It is important to show that most of the functionality of the book example might apply to any other product. However, according to some context drivers like "product", the basic scenario must be customized with respect to choreography and exchanged document types. If appropriate, we

show the tourism counterpart of the book example as well as a corresponding generic example with context driver notations.

4.1 Business Domain View

In our example the stakeholders in the business domain of book selling are interviewed. Thus, book retailers and book wholesalers amongst others (which we do not concentrate on here) describe their business processes in the domain under consideration. Business processes are recorded using the UML concept of use cases. Fig. 1 depicts on the left side use cases that show some sample business processes important to the retailer and on the right side those that are important to the wholesaler. The details of the business processes manifested in the associated use case descriptions and optional activity diagrams are not presented due to space limitations.

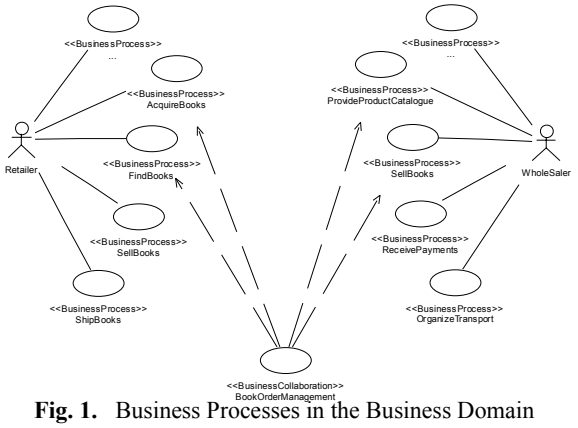

Fig. 1. Business Processes in the Business Domain

4.2 Business Requirements View

Considering the business processes of our example as depicted in Fig. 1, a business collaboration for *book order management* between a retailer and a wholesaler is established. This business collaboration is aligned with the business processes *find books* and *acquire books* on the retailer's side and *provide book catalog* and *sell books* on the wholesaler's side. Thus, dependency relationships are created from the *book order management* collaboration at the bottom of Fig. 1 to these processes.

The stakeholders' agreed shared view on the business collaboration book order management is documented in a corresponding use case description as depicted in Table 1.

According to the description of the business collabo-

Table 1: Book Order Management Description

Form: Business Collaboration	
Business Collaboration Name	Book Order Management
Description	A Retailer orders books from a wholesaler: Its up to the retailer to search books according certain criteria. The retailer gets back a list of corresponding books including price quotes. It is possible to put books on a reservation list at the wholesaler (books from this list can be canceled without charges). The retailer can order books either from the result list or from the reservation-list. The retailer must be registered for both making purchase orders and reserving books.
Preconditions	none
Begins When	Either the retailer searches for books to be ordered OR wants to view his current status of the reservation list
Ends When	When the wholesaler accepts the order and a contract is established;
Exceptions	Retailer finally does not want to place an order; Wholesaler does not accept an order; Books are not available; Wholesaler does not register the customer
Postconditions	Order completed
Initiating Partner Type	Retailer
Responding/Receiving Partner Type	Wholesaler
Initiating Events	Retailer's need for books
Terminating Events	Books are ordered
Scope	Search books in a catalog; manage books on a reservation list, customer registration process; establishing contracts for book orders;
Boundary	Managing purchase orders between retailer and wholesaler; it is limited to their communication and does not involve any communications with banks concerning payment or transporters concerning shipment;
Constraints	Business Collaboration is defined in the context of Books
Supporting Business Transactions and Business Collaborations	Register Retailer; Search Books; Reserve Books; Order Books; Present Reserved Books

ration, the book order management involves the registration of the retailer, the search for books, the reservation of books, the presentation of reserved books, and the ordering of books. These represent business collaboration use cases that are included in the book order management. Consider now an order management for tourism products. A use case description for tourism product order management is very similar to that for books. It differs only in the type of product and the initiating partner type that is a travel agency. The tourism product order management includes also very similar business collaboration use cases to that of book order management.

The similarity between book and tourism product order management mentioned above might be viewed as two different realizations of the same generic collaboration in two different contexts. This fact is presented in Fig. 2. We define a generic business collaboration use case *order management* which is performed by the actors *customer* and *seller*. It includes the business collaboration use cases *register retailer*, *search product*, *reserve product*, *present reserved products*, and *order products*. It should be noted that a business collaboration use case is defined in the UMM as the abstract generalization for business collaboration protocol use case and business transaction use case. We use the general term since at this point of the UMM process we do not know which of the two specializations applies. However, it will turn out later that all of the included use cases are business transaction use cases. Therefore, the corresponding stereotype is used in Fig. 2. In the *book order management* realization the *retailer* takes on the role of the *customer* and the *wholesaler* is in the role of the *seller*. Similarly, the *travel agency* takes on the role of the *customer* and the *wholesaler* is the *seller* in the *tourism product order management* realization.

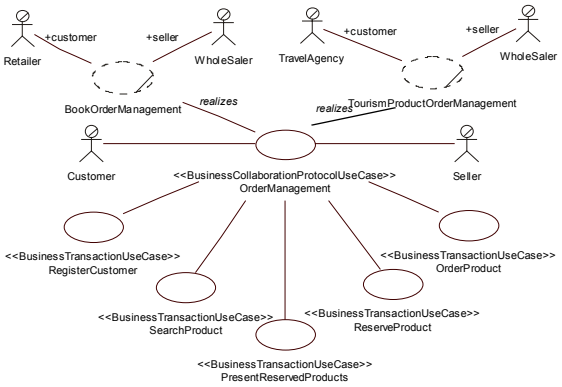

Fig. 2. Order Management Use Cases

4.3 Business Transaction View

The business collaboration protocol for our book order management example is depicted in Fig. 3. The choreography follows a description provided in the use case for the business collaboration realization. The book order management either begins by a search for books or by the query for the reservation list. After a search it is possible to order or reserve some books. Both activities require the retailer to be registered. If the result of a search was not satisfying another search is performed or the reserved books are queried. After a reservation of books was performed the next activity is either a new search or the query for the reserved books. Note that a query for books works for registered customers only, because otherwise they were not able to make a reservation. After querying the reserved books, some of these books might be ordered. The other choice is to perform a new search for books. The business collaboration always ends after ordering books. However, the search for books, the reservation, and the presentation of the reserved books might also be the last activity with the consequence that no books are ordered. Furthermore, we assume a maximum performance time of 24 hours for each business transaction activity and non of these being concurrent.

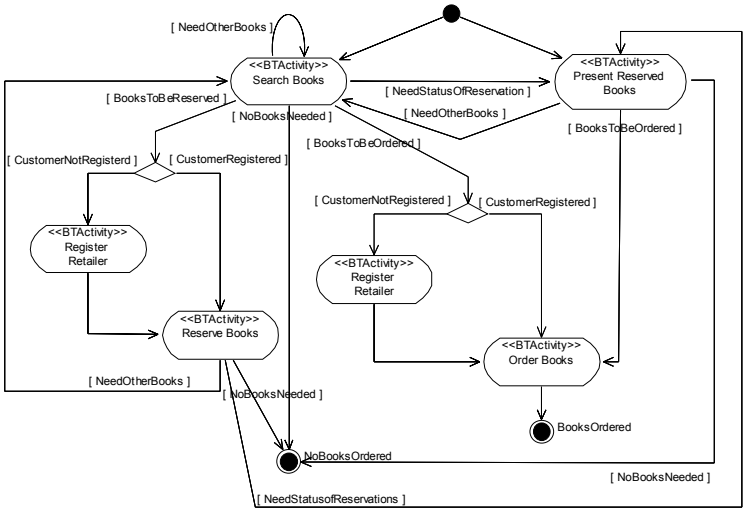

Fig. 3. Business Collaboration Protocol for Book Order Management

In Fig. 4 we present the business collaboration protocol for our tourism product order management example. Again, the choreography must follow a detailed description captured in the business collaboration realization. In addition to the different type of product, we assume the following differences compared to the choreography of the book order management: The reservation list is only valid during the business collaboration, it does not survive the end of the business collaboration. Thus, it does not make sense to start the business collaboration with a query for the reserved tourism products. Furthermore, it is not possible to order the tourism products immediately after having performed the search. First, all products for a trip must be reserved. Orders include only products appearing on the reservation list.

Since the two business collaboration protocols are quite similar they share a common business collaboration protocol for our order management example. It must be noted that this fact was already discovered at the business requirements view, when we defined the book order management and tourism product order management to be realizations of the generic order management. Fig. 5 presents the common business collab-

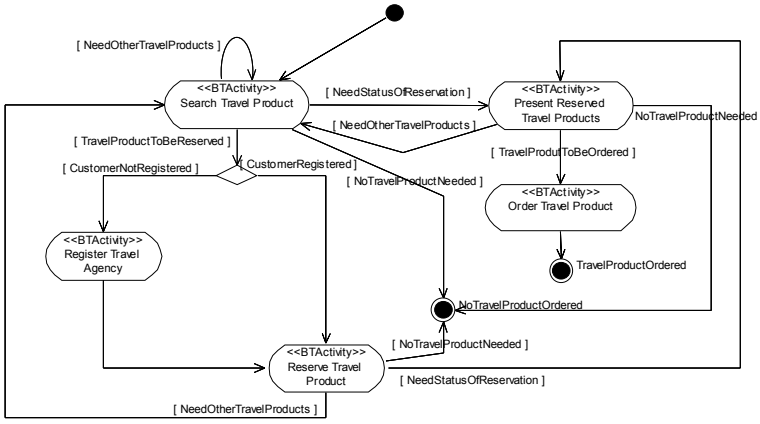

Fig. 4. Business Collaboration Protocol for Book Order Management

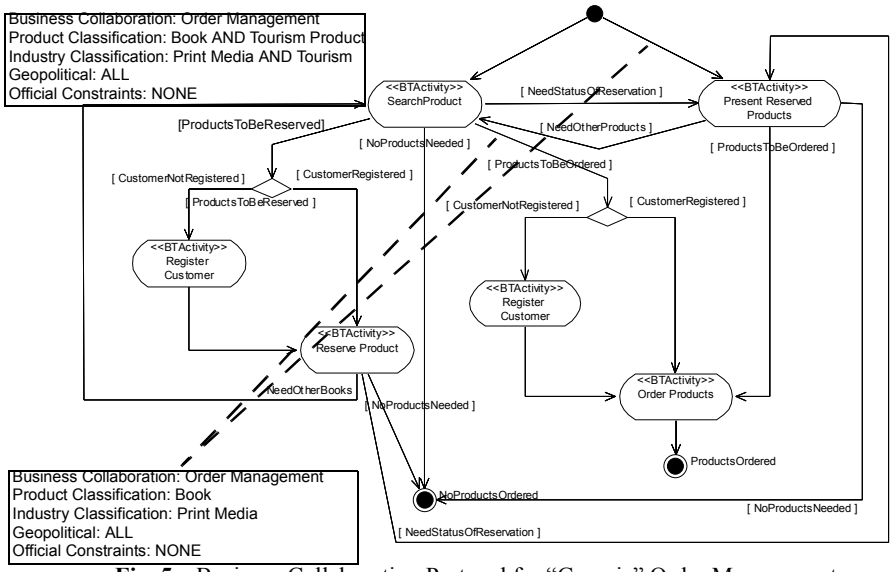

Fig. 5. Business Collaboration Protocol for "Generic" Order Management

oration protocol. When modeling with UMM, all the resulting models reflect the business environment the models are produced for. This business environment is characterized by different context drivers.

Our example is modeled in the context of the order management business collaboration, books and tourism products as product types, and print media and tourism as industries. No other restrictions apply on the other context drivers. This fact is denoted by assigning the context drivers to the business collaboration protocol. In Fig. 5 we present the context drivers in the upper left corner. However, according to the requirements the transitions from *start state* to *present reserved books* and from *search product* to *order product* is not valid in the tourism case. Thus, the transitions are not available in the default context, but in a context limited to books as products and print media as industry. The context drivers shown in the lower left corner of Fig. 5 are assigned to these transitions.

In the next step each of the business transaction activities is detailed by its own activity graph called business transaction. We present the business transaction *search product* in Fig. 6. As defined in the context drivers in the upper left corner of Fig. 6, the choreography for search product applies for both domains in our example. Since there is a response that does not immediately result in a contractual obligation and the responder has the information (about the products) already available, we selected the query/response pattern. The customer performs *request a search* as requesting activity, which is stereotyped according to the selected pattern. The responding activity *perform search* is executed by the seller. Since no obligations apply, no acknowledgments are needed and non-repudiation does not apply. Authorization is not required, because anyone is allowed to perform the search. We set the performance time of both the requesting and the responding business activity to 4 hours. Furthermore, the requesting business activity will retry to initiate the search 3 times in case of a control failure. The corresponding instantiation of the tagged values is denoted in Fig. 6 by comments assigned to the activities.

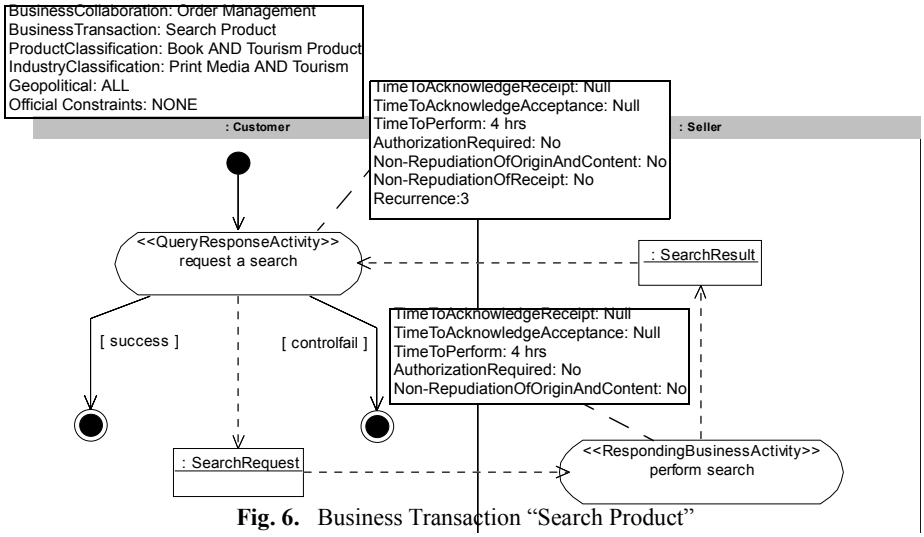

Fig. 6. Business Transaction "Search Product"

The requesting activity produces a *search request* document that is input to the responding activity. The responding activity outputs the *search result* document that is returned to the requesting activity. The structure of both document types is defined in class diagrams. In the following we concentrate on the *search result* document structure as an example. Table 2 presents a worksheet for the content description of the search result in our book example.

The worksheet to describe the document structure starts off with a general description of the document. It is the goal to map this description to a class diagram that is based on re-usable core components. Thus, the lower part of the worksheet is used to identify the core components used in the *research result* document. Unfortunately, there does not yet exist a globally agreed set of core components. Hence, the core components listed in Table 2 are still based on a pre-mature catalog developed for ebXML demonstration purposes in November 2001 [14]. Each row lists a basic core component. The representation term of a basic core component follows one out of 16 core component types. Core component types consist of a content component that carries the actual content plus supplementary components that give extra definition to the content. E.g., the ISBN number will go in the content component of an identifier type, whereas the fact that the identifier is of type ISBN goes into the supplementary one. Similarly, the quoted price itself instantiates the content component of an amount type, and the currency is denoted in a supplementary component. Furthermore, each basic core component is assigned to a so-called object-class. This gives a hint to which common business object (= class) the core component will be assigned to as an attribute. The business term provides a possible alternative term for the core component that is usually used by business people in the given context. The semantic description helps to further describe the core component.

Table 2: Document Description for the Research Results Document

Form: Content Description				
Content Description Name	SearchResult			
Content Description	A search result document is identified by a unique id assigned by the seller. Additionally, the search result document includes its creation date and time. The search result document returns a list of numbered items (books) that meet the search criteria. If no book matches the search criteria the list will be empty. Each item on the returned list refers to exactly one book. Furthermore, a price quote is made for each item. A book is described by the ISBN as unique identification, a title and its author(s).			
Attribute	**Representation Term**	**Object Class**	**Business Term**	**Semantic Description**
DocumentID	IdentifierType	SearchResult	Search Result ID	Unique id of the search result doc.
DocumentCreationDate	DateTimeType	SearchResult	Search Result Date	Creation date of the search result doc.
LineIdentifier	IdentifierType	LineItem	Line Number	Sequence number of the line item
ProductServiceDescriptionText	TextType	ProductService	Title	Title of the book
ProductServiceIdentifier	IdentifierType	ProductService	ISBN	ISBN of the book
Author	TextType	BookItem	Author	Author(s) of the book
UnitChargePriceAmount	AmountType	UnitChargePrice	Price Quote	Price quote for the corresponding book

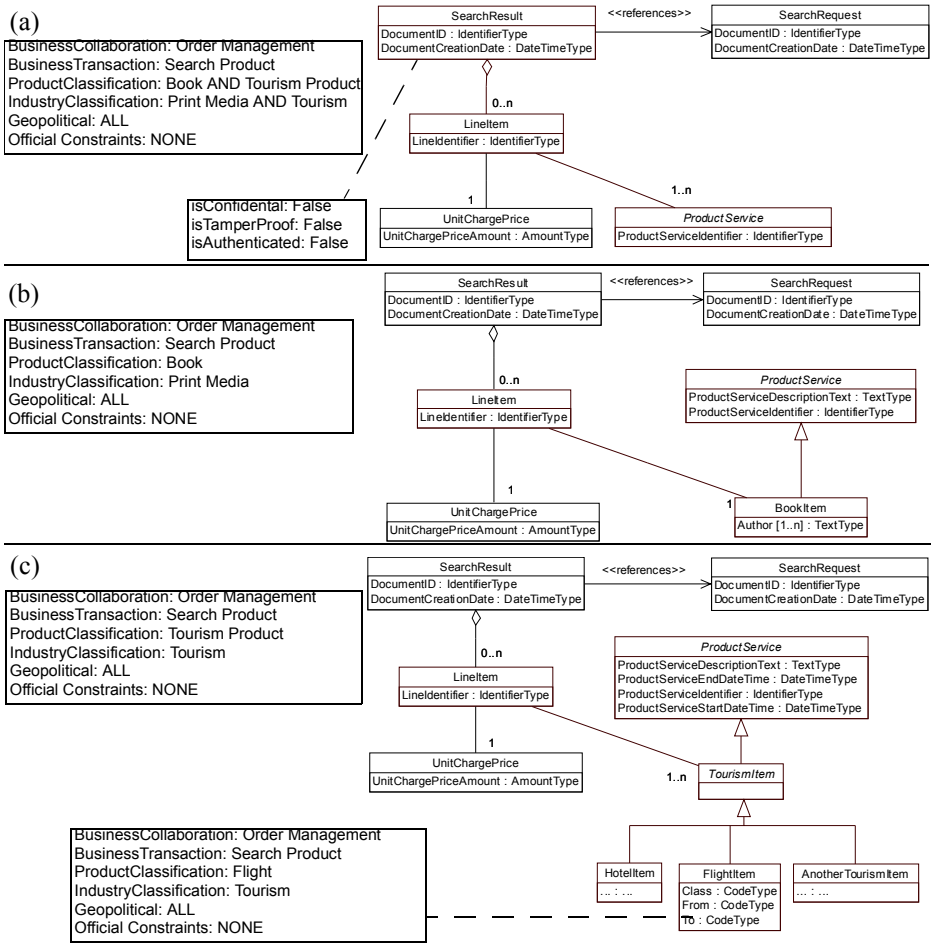

Fig. 7. Search Result Document: (a) Common, (b) Book, (c) Toursim Product

Given the document description in Table 2 it is easy to recognize that some core components are specific to a search result for books and others will appear in a result list for any product. This becomes even more evident by the corresponding description of the tourism product example, which is not lined out due to space limitations. Given a significant overlay between the search result documents of our two examples, it is possible to create a class diagram for the common structures. This class diagram is depicted in Fig. 7a. The context drivers for the common search document are defined in the box left to the diagram. The tagged values in the comment assigned to the class *search result* denotes the fact that neither the search result document nor one of its components are confidential, tamper proof, or authorized. For reasons of simplicity, we define this flags only on the document level. Commonly, the *search result* is an aggregation of zero to more *line items*. Each *line item* refers to one or more *product/service* and exactly one *unit charge price*. The types of the attributes correspond to core component types. The *search result* class covers the unique *document id* assigned by the seller and the *document creation date*. The *line identifier* corresponds to the sequence number of the *line item* in the result list. The *unit charge price amount* attribute of class *unit charge price* is self explanatory.

The structure of each *product/service* in the result list depends on the context of the product, and must hence be replaced by a specialized item in a given context. It follows that the class *product/service* in the common class diagram is abstract. A *product/service identifier* is common to all specialized items and is thus an attribute of the generalized class. In Fig. 7b we present the search result structure for the book example. Given the context drivers on the left, the abstract *product/service* class is replaced by a specialized *book item*. *Book item* inherits the *product/service identifier* and the *product/service description text* for its title. Furthermore it defines *author* as an additional attribute. Fig. 7c shows the search result structure in the tourism domain. In this case specializing the *product/service* as a *tourism item* is still too general. Since the structure of a *tourism item* varies from one tourism product type to another, a further specialization for different tourism product types is required. In the example depicted in Fig. 7c the context drivers for *flight item*, *hotel item*, etc., follow the product type.

4.4 Business Service View

During the business service view the message exchanges between the network components are defined. In our example there exist two network components: one for the customer service and one for the seller service. Each business transaction of the BTV maps to a corresponding service transaction in the BSV. Fig. 8 depicts the *search product* service transaction. The corresponding business transaction does not require any acknowledgments. Therefore, the resulting sequence diagram is quite simple and only includes document exchanges. The customer service calls the *performSearch* operation of the seller service by sending the search request document. The seller service returns the search result document by calling the *requestASearch* operation that initiated the service transaction.

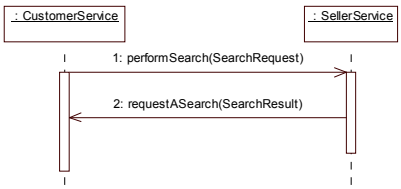

Fig.8. Service Transaction Search Product

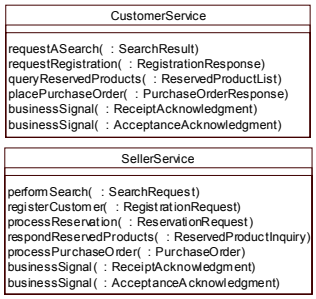

Fig. 9. Network Components

By mapping all five business transactions of our example to service transactions, we exactly define the services to be supported by each network component. Fig. 9 shows the resulting network components and their services. Network components are represented as interface classes and their services as operations. Software vendors supporting a role in the simple order management must implement the corresponding interface class in their applications. The unambiguous input to a service was defined in the class diagram of the BTV. The collaboration protocol (Fig. 5) defines the choreography among transactions, whereas the sequence diagram of the BSV (Fig. 8) defines the choreography within a transaction.

5 Summary

In this paper we present UN/CEFACT's modeling methodology (UMM). The goal of UMM is to capture the business knowledge that enables the development of low cost software components by software vendors to help the small- and medium-size companies. The software process-like methodology is based on 4 workflows: (1) The business domain workflow elicits and organizes business processes and information in the business-to-business domain. This includes the discovery of existing business processes of importance to stakeholders and their categorization. (2) The business requirements workflow uses the discovered business process to identify possible collaborations. Furthermore, the workflow includes the creation of detailed requirements for the business collaboration documented in UMM use case diagrams. (3) The business transaction workflow further elaborates the requirement use cases by detailing the activities that occur and defining their choreography in activity diagrams. Furthermore, the structure of business documents exchanged in business transactions is modeled in class diagrams. (4) The business service workflow describes the business collaborations amongst network components. The network components are described as interface classes that are integrated by software vendors into their products. The workflow precisely defines the dynamics involved and the operations necessary to support a role in the collaboration.

All the workflows are detailed by means of examples on ordering/selling books and tourism products. In these collaborations the choreography as well as the document structures are very similar, but differ in some details. In order to avoid a proliferation of similar business collaborations, a method unambiguously adapting the common collaboration to the specific business environment is needed. Thus, we have extended the current version of UMM by the concept of context drivers. This allows software reuse for the common parts of a collaboration and unambiguous, context-specific adaptations. Of course, it is recognized that even with UMM, the issue of businesses doing things differently would not disappear. However, in regard to the SMEs, it is envisioned that software providers would create applications that implement the most popular scenarios.

The example introduced in this paper will serve as reference example for our future work items. Topics of interest include a more formal definition of the context drivers. Appropriate code lists, e.g. UN/SPSC for products and services, must be identified to ensure clear semantic definitions and machine-processable context drivers. Furthermore the context drivers assigned to UMM stereotypes must be machine processable. The object constraint language (OCL) seems to be a candidate for this purpose. Additionally, we are working on the transformation of the UMM artifacts into current B2B technologies, such as ebXML [3,7] and Web Services [20].

6 References

1. Booch, G., Jacobson, I., Rumbaugh J.: The Unified Modeling Language User Guide. Addison Wesley Object Technology Series, Reading, (1998)
2. Dogac, A., Tambag, Y., Pembecioglu, P., Pektas, S., Laleci, G. B., Kurt, G., Toprak, S., Kabak, Y.: An ebXML Infrastructure Implementation through UDDI Registries and RosettaNet PIPs. ACM SIGMOD International Conference on Management of Data, (2002)
3. ebXML: Homepage of the ebXML Initiative. http://www.ebXML.org
4. ebXML: ebXML Technical Architecture Specification v1.04, (2001) http://www.ebxml.org/specs/ebTA.pdf
5. Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF. http://informatik.uibk.ac.at/users/c70385/wese/wsmf.paper.pdf
6. 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, pp. 5-12, (1989)
7. Hofreiter, B., Huemer, C., Klas, W.: ebXML: Status, Research Issues and Obstacles, Proc. of 12th Int. Workshop on Research Issues on Data Engineering (RIDE02), San Jose (2002)
8. Huemer, C.: Defining Electronic Data Interchange Transactions with UML. Proceedings of HICSS-34, Maui, (2001)
9. Huemer, C.: <<DIR>>-XML² -Unambiguous Access to XML-based Business Documents in B2B E-Commerce. Proc. of 3rd ACM Conference on Electronic Commerce, Tampa (2001)
10. ISO: Open-edi Reference Model. ISO/IEC JTC 1/SC30 ISO Standard 14662 (1995)
11. Lee, R.M.: Documentary Petri Nets: A Modeling Representation for Electronic Trade Procedures. Business Process Managements - models, techniques and empirical studies. Springer LNCS Vol. 1806 (2000)
12. Lenz, K., Oberweis, A.: Interorganizational Business Process Management with XML Nets. Advances in Petri Nets. Springer LNCS Vol. 2472 (2003)
13. Ling, S., Loke S.W.: Advanced Petri Nets for Modelling Mobile Agent Enabled Interorganizational Workflows. Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS 2002), Lund (2002)
14. UN/CEFACT TMG: Catalog of Core Components. Version November 2001. http://webster.disa.org/cefact-groups/tmg/downloads/CCWG/drafts/CC-Catalogue_11_01.zip
15. UN/CEFACT TMG: UN/CEFACT Modelling Methodology. http://webster.disa.org/cefact-groups/tmg
16. UN/CEFACT TMG: UMM Meta Model. Revision 12. http://webster.disa.org/cefact-groups/tmg/downloads/BPWG/drafts/UMM-MM-V20030117.zip
17. UN/CEFACT TMG, Core Component Technical Specification, Version 1.90, (2002) http://webster.disa.org/cefact-groups/tmg/downloads/CCWG/for_review/CCTS_V_1pt90.zip
18. van der Aalst, W.M.P.: Modeling and Analyzing Interorganizational Workflows. Proc. of Int. Conf. on Application of Concurrency to System Design (CSD'98), Fukushima (1998)
19. van der Aalst, W.M.P.: Process-Oriented Architectures For Electronic Commerce and Interorganizational Workflow. Information Systems, Vol. 24, No. 8, Elsevier Sciences Ltd (2000)
20. W3C: Web Services Activity. http://www.w3.org/2002/ws/