Registering UML Models for Global and Local Choreographies

Birgit Hofreiter
Department of Information Systems, University of Technology Sydney, Australia
Faculty of Computer Science, University of Vienna, Austria
birgit.hofreiter@univie.ac.at

ABSTRACT

A local choreography describes how a business partner - from his perspective - interacts with other business partners. If local choreographies are developed in isolation, the local choreographies of different partners will not match. A global choreography, which describes the interactions from a neutral perspective, may serve as an agreement model between the partners and each partner may derive his local choreography. Global choreographies developed by standard organizations, industry consortia, or market leaders should be publicly available. Furthermore, business partners have to register their local choreographies and bind it to the supported global choreographies. In this paper we present an appropriate registry meta model that customizes the ebXML registry for the purpose of registering UML-based models for global and local choreographies and maintaining the dependencies between them.

1. MOTIVATION

In the 1990ies the work of Hammer and Champy on business process reengineering [5] attracted a lot of attention and companies started to rethink their business processes. Hammer and Champy define a business process as an organized group of related activities that together create customer value. It is the goal to optimize business processes in a way to meet a company's business goals, such as financial targets. Accordingly, the activities of a business process must be arranged to optimize a company's output. A business process model defines which activities are executed in which order under which conditions by whom and by using which resources.

Over the last few years supply chain management involving multiple parties became more and more popular, leading to an inter-organizational focus of business process management. In this context the above given definition of a business process of Hammer and Champy toward creating customer value does not fit anymore, because a supply chain includes many seller - customer relationships. An inter-organizational business process is an organized group of related activities carried out by multiple organization to accomplish a common business goal. An inter-organizational business process does not focus on the internal tasks of an individual organization, it rather focuses on the tasks carried out between the actors.

The different focus of intra- and inter-organizational business processes is also reflected in the discussions of Web Services used to implement the business processes. Orchestration and choreography describe closely related, but well distinguished concepts [24]. Orchestration deals with the sequence and conditions in which one business process calls its components to realize a business goal. Choreography describes business processes in a peer-to-peer collaboration. It describes the flow of interactions between the participating business partners that interlink their individual processes. We distinguish local and global choreographies. A local choreography describes the flow from a participating partner's point of view. It makes the public parts of its local process visible to others. A global choreography defines the inter-organizational process from a neutral perspective. According to their definitions, choreographies are relevant for the specification of inter-organizational processes, whereas orchestrations are used for the composition of internal processes.

Evidently, there exists a strong correlation between orchestrations, local, and global choreographies. A local choreography is a projection on a business partner's orchestration describing his external behavior. If two business partners want to collaborate they must have complementary local choreographies. This means that the flow of activities is the same, however each basic activity is reciprocal to the one in the other local choreographies, i.e. one partner invokes a service, whereas the other one receives a call of the same service. A global choreography describes the same flow of activities from a neutral perspective and keeps track who is the sender and who is the receiver for each activity. A global choreography has the potential to serve as a kind of contract on the flow of activities that is agreed between the business partners.

In order to set-up a collaboration between business partners one may start bottom-up from the orchestrations or top-down from the global choreographies. A bottom-up approach bears the limitation that if each partner develops its local choreography in isolation, it is rather unlikely that their processes are complementary to each other. Thus, it works only, if one partner dominates the partnership and the other one adapts his interfaces accordingly. In this case, discovering potential business partners requires complex comparisons of local choreographies. In a top-down approach, standards organizations, industry consortia or market leaders define a global choreography. An enterprise supporting a standard global choreography has to derive the local choreography of the supported role and bind its interfaces accordingly. The complexity of discovering potential business partners is reduced to finding partners supporting a complementary role in a global choreography.

In this paper we follow the top-down approach. We concentrate on bridging global and local choreographies in conceptual modeling. In particular, our approach considers UML-based profiles for modeling choreographies: UN/CEFACT's modeling methodology for global choreographies [27] and our attuned UML profile for local choreographies [6]. Our approach helps to develop local choreographies that are compliant to each other. This is guaranteed by the fact that each partner derives its local choreography consistently to a commonly agreed global choreography. This requires that the commonly agreed global choreographies are available in public registries. In order to facilitate an organization in finding appropriate global choreographies, a business-context sensitive registration mechanism to store and access the artifacts of global choreographies must be provided.

Even if project partners will use the proposed approach of deriving their local choreographies based on common global choreographies, it is not guaranteed that they will be able to do business electronically with each other. This is due to the fact that they may not be able to find each other. If the business partners just register their local choreographies, the fact that these are based on the same global choreography will be lost. Hence, it is important to maintain the link between local and global choreographies in the registry. In this paper we present the concepts necessary to store both global UMM choreographies and local choreographies in an ebXML registry and to maintain the relationships between them in the registry meta model.

The remainder of this paper is structured as follows: In section 2 we give a brief introduction of UMM [27] - of which we are co-editors - and its artifacts. In section 3 we give an introduction to our UML profile for local choreographies and orchestrations. In section 4 we outline how UMM models and its local choreographies and parts thereof are represented in our proposed registry model. Section 5 focuses on related work and a short summary in section 6 concludes the paper.

2. UMM: GLOBAL CHOREOGRAPHIES

UMM is used to model an inter-organizational business process concentrating on the flow of interactions between collaborating business partners. It does not address their private processes. The execution of an inter-organizational business process depends on commitments established between the participating partners. UMM is used to model the procedural and data exchange commitments that must be agreed upon between the business partners at design time of the inter-organizational business process, i.e. before executing the process. Inasmuch the UMM model becomes a kind of "contract" that guides the business partnership. Since most commitments are made between two partners only, a UMM model - such as most contracts - is agreed upon between two parties. Also similar to a contract, a UMM model describes the commitments on the information flow from a neutral perspective. It follows that UMM describes a bi-lateral and global choreography. The UML profile of the UMM has been defined for this special purpose.

The UMM follows a well-defined development process that produces a set of well-defined artifacts. The development process runs through three major phases, which correspond to the three top level packages of UMM: the *business requirements view*, the *business choreography view*, and the *business information view*. In this paper we do not concentrate on the development process and limit ourselves to only those artifacts that are relevant in the registry context. The reader interested in all details is referred to the UMM paper in these proceedings [X]. For the purpose of this paper we focus on artifacts of two sub-views of the *business choreography view*: the *business transaction view* and the *business collaboration view*. A *business transaction view* covers the artifacts of business transactions, which is a business information exchange between two partners including an optional response. A *business collaboration view* comprises the artifacts of a business collaboration, which spans over multiple business transactions.

In order to exemplify the UMM artifacts we use a simple `order from quote` example. In this example a buyer requests a quote from a seller. Once he receives it, he is able to order products, which is confirmed by an order response. Furthermore, the seller checks with his bank the creditworthiness of the seller prior to issuing a quote. Since UMM models are bi-lateral choreographies, this business case is split-up into two business collaborations. The `order from quote` business collaboration between the buyer and the seller consists of two business collaborations: `request for quote` and `place order`. Figure 1 depicts the corresponding use cases. The `check credit` business collaboration between the seller and the bank includes only one homonymously named business transaction, which use cases are not depicted here.

A UMM *business transaction view* is characterized as follows:

- A *business transaction view* includes exactly one *business transaction use case* and the two *authorized roles* participating in this use case.
- A *business transaction use case* is the parent of exactly one *business transaction* that is modeled by an activity diagram.

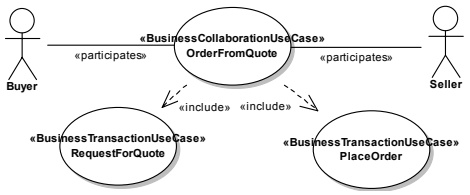

Figure 1: Business Collaboration/Transaction Use Cases

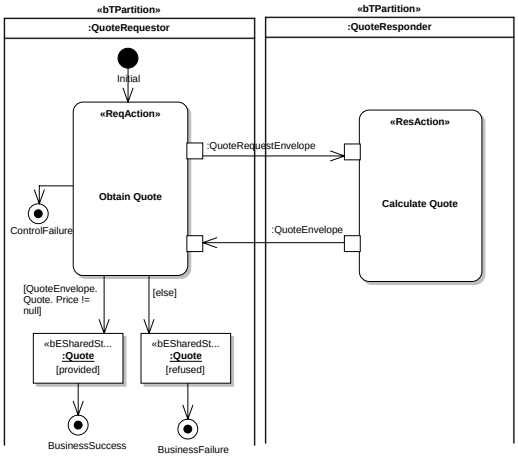

Figure 2: Business Transaction: Request For Quote

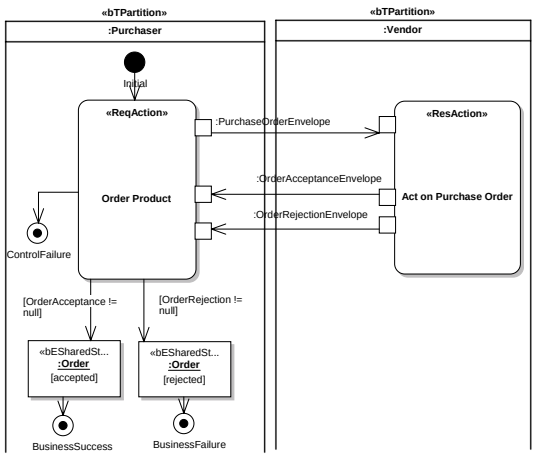

Figure 3: Business Transaction: Place Order

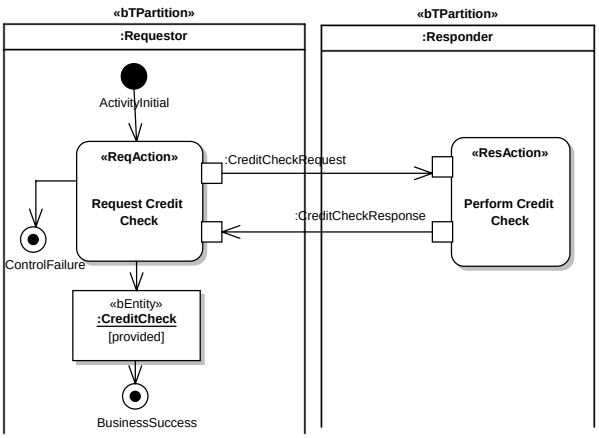

Figure 4: Business Transaction: Check Credit

- A *business transaction* is built by two *partitions*, i.e. one for each participating *authorized role*.
- Each *partition* includes exactly one business action, either a *requesting business action* or a *responding business action*.
- Exactly one *requesting business information envelope* is exchanged from the *requesting business action* to the *responding business action*.
- Zero, one, or one out of more alternative *responding business information envelopes* is returned from the *responding business action* to the *requesting business action*.
- The *requesting business action* leads to one of more alternative *business entity states* depending on the received results.

Our example involves three *business transaction views*. The first one, `request for quote`, comprises the *business transaction use case* `request for quote` - depicted on the lower left of figure 1 - which is the parent of the *business transaction* in figure 2. This *business transaction* follows the pattern described in the bullet list above. A `quote requestor` performs `obtain quote` which sends a `quote request envelope` to the `calculate quote` action of the `quote responder`. A `quote envelope` is later returned to the `obtain quote` action. Depending on the result - whether a price is given in the quote envelope or not - the *business transaction* leads to one of the two *states* of the *business entity* `quote`: `provided` or `refused`. The second *business transaction view*, `place order`, covers the *business transaction use case* `place order` - depicted on the lower right of figure 1 - which is the parent of the *business transaction* in figure 3. Again it follows the pattern, however, in this case multiple alternative *responding business envelopes* - one for acceptance and one for rejection - are involved. Finally, the *business transaction view* `check credit` includes an homonymously named *business transaction use case* and its underlying *business transaction* depicted in figure 4.

A UMM *business collaboration view* is characterized as follows:

- A *business collaboration view* includes exactly one *business collaboration use case* and the *authorized roles* participating in this use case.
- A *business collaboration use case* includes other *business collaboration use cases* and/or *business transactions cases*, each defined in its own *business transaction view*.
- A *business collaboration use case* is the parent of exactly one *business collaboration protocol* that is modeled by an activity diagram.
- A *business collaboration protocol* is built by *business transaction actions* and *business collaboration actions*. The former calls a *business transaction*. The latter calls another *business collaboration protocol*.
- A *business collaboration protocol* includes a *partition* for each participating *authorized role*.
- Information flows - the *init-flow* and the optional *re-flow* - connects each *business transaction action* with the *partitions* in order to denote who is playing which role in the underlying *business transaction*.
- If an *init-flow* may start a *nested collaboration* - defined in another *business collaboration view* - the *init-flow* does not lead to a *partition*, but targets a *nested collaboration* placed in this *partition*.

In our example, the first *business collaboration view* includes the *business collaboration use case* `order from quote` depicted on top of figure 1. This use case is the parent of the *business collaboration protocol* of figure 5. This *business collaboration protocol* includes two *business transaction actions*. The first one calls the *business transaction* `request for quote`. If this *business transaction* sets the *business entity* `quote` to state `refused`, the *business collaboration protocol* ends with a failure. If it sets it to `provided`, the *business collaboration protocol* continues with a call of the *business transaction* `place order`. Depending on whether the `order` is set to `accepted` or `rejected`, the *business collaboration protocol* ends with a success or a failure. The *init-flow* starts for both *business transaction actions* from the *buyer*. Thereby, it is denoted that the buyer is the initiator of the underlying *business transactions*, which is the `quote requestor` in the first transaction and the `purchaser` in the second transaction. The *init-flow* of the `request for quote` *business transaction* targets the *nested collaboration* `check credit`. This hints to the fact that the receiver of the request, i.e. the `seller`, has to perform the *nested*

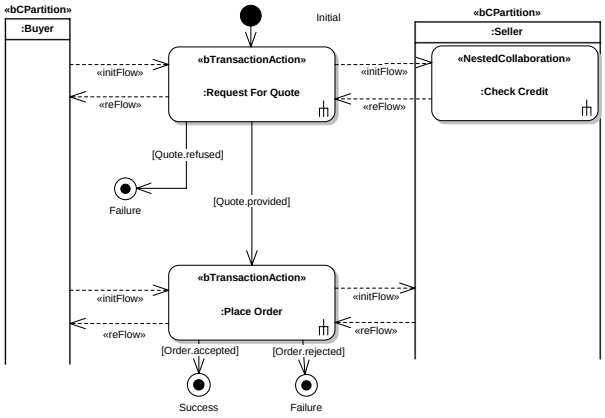

Figure 5: Business Collaboration Protocol: Order from Quote

collaboration before replying to the initiator, i.e. the `buyer`. The *nested collaboration* `check credit` is defined in its own *business collaboration view*. Since the *business collaboration protocol* `check credit` is rather simple, consisting of a single *business transaction action*, we omit to depict it.

On first sight, one may argue that it would be more convenient to create just a single *business collaboration protocol* that calls all three `business transactions`. However, this is not possible due to the *business transaction* semantics: a *business transaction* has to complete before another one starts. Since `request for quote` starts before `check credit`, but cannot complete before it, this approach is impossible. Therefore, UMM is dedicated to bilateral collaborations. A detailed discussion on this topic is given in [6].

3. LOCAL CHOREOGRAPHIES AND ORCHESTRATIONS

Usually, UMM is used to develop industry reference models. If a business partner decides to support a certain role of a UMM model, it is envisioned that she is able to collaborate with any other party supporting the complementary role in the same business process. Accordingly, UMM may help finding appropriate business partners. However, if a business partner performs an analysis and design process for its inter-organizational business processes, it is not straight forward to re-use existing UMM models. Usually, a certain company or organization will describe its business processes from its own local view and not from a global view. Furthermore, a company usually has a predecessor and one or more successors along a supply chain when executing a certain function in that chain. Consequently, the party under consideration will communicate with multiple other parties during the lifetime of its inter-organizational business process. It follows that an analysis and design process for a B2B project of a certain partner result in models that describe multi-party local choreographies or orchestrations, respectively.

In this section we present a UML profile (i) that is able to model local choreographies or orchestrations and (ii) that has a dedicated binding to UMM allowing a straight-through modeling approach. This profile has the following main characteristics which are discussed in detail in [6]:

- A *local process* models a local choreography or an orchestration
- The main building blocks of a *local process* are *initiating activities* and *reacting activities*. An *initiating activity* is used to model the inside of a UMM *requesting business activity*. Similarly an *reacting activity* models the inside of a UMM *responding business activity*.
- The transitions between *initiating/reacting activities* must correspond - including their guards - to the transitions in the *business collaboration protocol* from which the *local business activities* are derived.
- The *initiating/reacting activities* send and/or receive *information envelopes*. Thus, an object node is added to an *initiating/reacting activity* for each incoming and outgoing *information envelope*.
- A *receive business information* - which is a UML *accept event action* - recognizes the event of an incoming *information envelope* and delivers it to the input node of the corresponding *initiating/reacting activity*.
- The flow within a *local activity* comprises the following concepts: *send business information*, *call nested transaction*, *private actions*, and *private activities*
- A *send business information* - a UML *send signal action* - delivers an *information envelope* to the output node of the *local activity*.
- A *call nested transaction* - a UML *call action behavior* - calls a flow in another *initiating/reacting activities* of another *local process*.
- *Private actions* and *private activities* model tasks internal to the organization and are not visible to other parties. *Private activities* may be decomposed, *private actions* must not. *Private actions* and *private activities* are used to model orchestrations, and must not be used in local choreographies.

Figure 6 presents the local process of the seller in our `order from quote` example who also performs `check credit` with the `bank`.

The UMM *business collaboration protocol* for `order from quote` (figure 5) consists of a sequence of `request for quote` and `place order`. In the underlying *business transactions* the seller performs the *responding business action* `calculate quote` (figure 2) and `act on purchase order` (figure 3), respectively. If follows that the *local process* (figure 6) includes a sequence of two *reacting activities* called `calculate quote` and `act on purchase order`. The flow between them and the guard conditions are derived from the *business collaboration protocol* (figure 5). This means, a refused quote leads to a failure state. The transition from `calculate quote` to `act on purchase order` is guarded by the fact that a quote was provided. A rejected order leads to a failure state after `act on purchase order`, whereas an accepted order leads to a success.

The next step is detailing each of the *reacting activities* in the *local process*. An object node is added for each incoming and outgoing *information envelope*. In our example, `calculate quote` and `act on purchase order` both have an input and an output node. The *information envelopes* assigned to these nodes correspond to the input and output of the corresponding *requesting/responding*

business activities. From figure 2 it follows that `calculate quote` has an input of `quote request` and an output of `quote response`. Similarly, `act on purchase order` (figure 3) receives an input of `purchase order` and outputs an `order response`.

For each input node we add a *receive business information*. It is used to recognize the event of an incoming *information envelope*. In case of a *reacting activity* this event and the resulting transfer of the *information envelope* is required to start the *reacting activity*. In our example of figure 6 the overall initial state leads immediately to the `calculate quote` activity. However, before the first activity within `calculate quote` is started, the *receive business information* `exchange quote request` must recognize the receipt of a `quote request` and transfer it to `calculate quote`.

In order to demonstrate modeling the flow within an *initiating/reacting activity* we take a look on `calculate quote` in figure 6. As mentioned above it is started with an incoming `quote request`. So the first task of the flow is `file quote request`. The nested task `request credit check` from a bank is the next one. Once this is done the flow continues with `act on the credit check results`. Logically, the next step is either `provide quote` or `refuse quote`. In either case `exchange quote response` is the last task. Note, if a quote is refused the `quote response` will state the reason of rejection.

It is easy to recognize that the activities with an *initiating/reacting activity* are based on different stereotypes. Most of them are *private actions* which are only used when modeling orchestrations. The tasks `request credit check` and `exchange quote response` are also relevant for local choreographies. The latter is of stereotype *send business information* which is a special kind of the UML *send signal action*. In our example, `exchange quote response` transfers the `quote` to the output node of `calculate quote`. The fact, that the `quote` is returned to `obtain quote` (executed by the buyer) is not shown in the local process - it is already defined in the `request for quote` *business transaction* (figure 2).

`Check credit` is of stereotype *call nested transaction*. It enables nested transactions with third parties. In our example, checking the customer credit has do be done after receiving a `quote request` and before responding to it. Accordingly, we define the concept of a *call nested transaction* as a special kind of the UML *call action behavior*, used to call another structured activity - which is an *initiating activity* of the same party. This means, the `calculate quote` activity includes the *call nested transaction* `request credit check`. This one calls the synonymously named *initiating activity request* `credit check` - which is the seller's task in the *business transaction* `check credit` (figure 4).

We again specify a flow within the *initiating activity* `request credit check`. After finishing this flow, control is given back to `calculate quote`. Since `request credit check` is an *initiating activity*, its internal flow first includes a *send business information* action before receiving a return back. However, its flow is only able to continue with `file credit check`, if a `credit check response` is recognized by the *receive business information action* `exchange credit check response`.

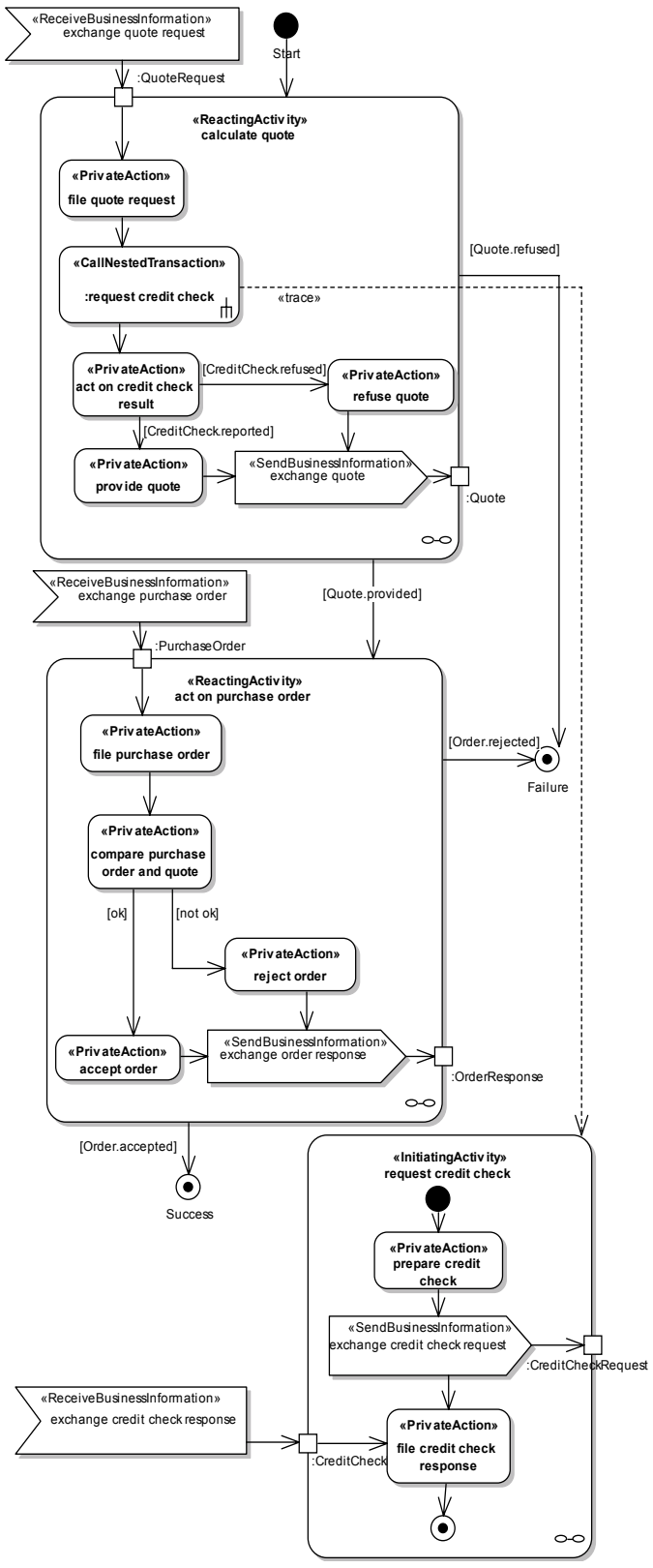

Figure 6: Local Process of the Seller

4. REGISTRY SUPPORT

In the previous subsections we concentrated on the modeling of global choreographies, local choreographies and orchestrations. Whereas orchestrations include process steps that have to be kept as an or-

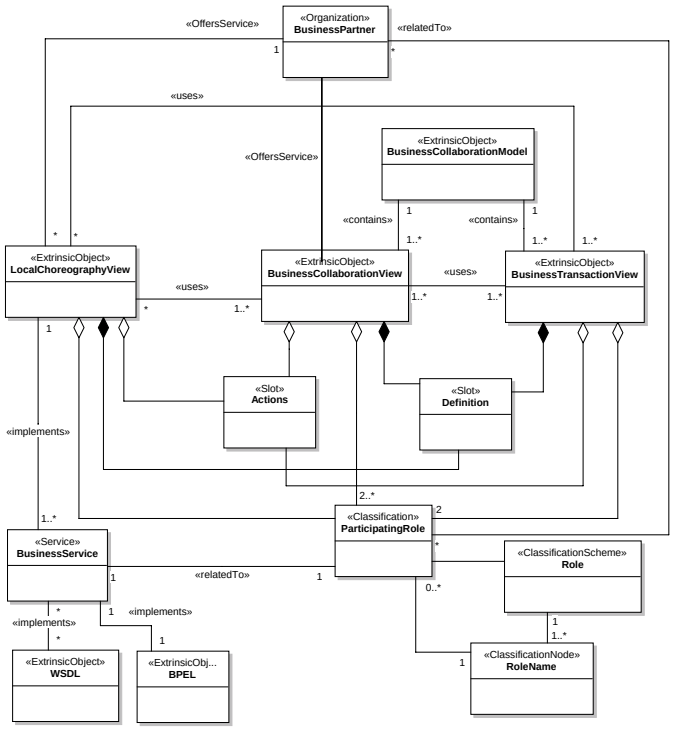

Figure 7: Registry Meta Model

ganization's secret, global and local choreographies must be announced to the public in order to facilitate business collaborations. Hence, the UMM models and the local choreographies - which must not use the private actions/activities - have to be available in a registry. This registry has to support the following typical, not exhaustive list of queries:

- What global choreographies are available?
- Which global choreographies are supported in which role by a certain company?
- What is the local choreography of a certain company supporting a certain role in a certain global choreography?
- Which global/local choreographies include a certain business process step (action)?
- Which companies support the same global choreography as myself, but in a complementary role, and what is their local choreography in this context?
- Are there any other choreographies the make use of the same or a similar set of actions that I support in my current set of choreographies?

It is our goal to store the global and local choreographies in a registry and make them publicly available. For this purpose we use an ebXML registry, which allows us to customize its registry information model for our special purpose. In an ebXML registry the artifacts are stored as extrinsic objects. An extrinsic object may be of any XML-based format, any other text-based format, or even a binary. Since we use UML as language to represent the global and local choreographies, we prefer to use the XML Metadata Interchange (XMI), which is commonly used for the serialization of UML models. The content of an extrinsic object is encapsulated. In other words, the registry does not care about its content. Accordingly, queries to the registry do address the inside of an extrinsic object. A query, rather, selects those extrinsic objects which are associated with the meta data matching the request. It follows that the design of a customized ebXML registry for maintaining global and local choreographies has to address the following points: (i) Define an appropriate granularity of extrinsic objects that serve as self-contained objects. (ii) Establish well-defined relationships between extrinsic objects that allow traces between them. (iii) Identify appropriate meta data for extrinsic objects that allow an effective search. The resulting registry meta model extending the ebXML registry information model (RIM) is depicted in figure 7.

The first problem we address is the granularity of the modeling artifacts to be stored as extrinsic objects. Usually, the choreographies are developed within a UML modeling tool. The resulting UML model covers all artifacts created during a UMM development process - this means a lot of artifacts created during the requirements elicitation phase (which are not discussed here), artifacts from one or many *business choreography views*, and artifacts from several *business transaction views*. In addition, the same UML model may include artifacts from one or more *local choreographies*. It follows, that all these artifacts are represented within one single UML file of the tool, which may also be presented as a single XMI file. This XML file is a candidate to be managed as an extrinsic object, but it is questionable whether it provides an appropriate granularity or not. On the one hand side, full models may be worth registering, because designers of to-be-developed choreographies may be interested in getting the full information on existing ones in compact form in order to re-use/customize it to their own needs. On the other hand side, business partners searching the registry may only be interested in parts of the model, e.g. in one *business collaboration protocol* of a particular *business collaboration view*, but not in other ones. In order to fulfill both requirements, it is suggested to allow the registration of full models as well as of well-defined parts thereof. Accordingly, consider the following types of extrinsic objects in our registration approach:

- Business Collaboration Model
- Business Collaboration View
- Business Transaction View
- Local Choreography View

The ebXML registry information model (ebRIM) offers an extension mechanism to create subtypes of extrinsic object. Accordingly, we create a subtype for each of the above mentioned types. A *business collaboration model* includes the full model. A *business collaboration view* contains a *business collaboration protocol* and the participating *authorized roles*. Similarly, the *business transaction view* covers a *business transaction* and the two *authorized roles* participating in it. The *local choreography view* covers the local process (not including *private actions/activities*) and the *authorized role* executing the local choreography.

Usually, a UML tool allows exporting parts of a UML model to an XMI fragment. Thereby, XMI fragments exactly matching the needs of the different subtypes of extrinsic objects may be created

by hand. However, if a full model is submitted to the registry, it is not sufficient to store it simply as an extrinsic object. Evidently, the full model includes amongst other artifacts also the ones of *business choreography views*, *business transaction views*, and/or *local choreography views*. Since users may explicitly just search for these parts of a model, it is necessary to store them as well as extrinsic objects of the corresponding subtype. Thus, the registry has to provide a mechanism to extract the relevant XMI fragments from the XMI file of a full model.

Furthermore, the registry management has to ensure consistent results. A *business collaboration protocol* (of the *business collaboration view*) includes *business transaction actions* that call *business transactions* (of the *business transaction view*). Thus, if only a *business collaboration view* is submitted, it is only accepted in case all required *business transaction views* are already registered. Similarly, a local process is based on one or more *business collaboration views*. Hence, a submitted *local choreography view* is accepted, if all required *business collaboration views* are already available.

The discussion on the dependencies between the artifacts do not only have a consequence on the registry management, but also on the registry meta model. The relationships should not only exist in the encapsulated content of the extrinsic objects, but they also have to be expressed as associations between the extrinsic objects, because users may be interested in follow the traces. The ebXML registry information model predefines a number of association types which fit the needs of our purpose. Accordingly, we define a *contains* association from *business collaboration model* to each of *business collaboration view*, *business transaction view* and *local choreography view*. In addition, we specify a *uses* association from the *business collaboration view* to the *business transaction view*, as well as another one from the *local choreography view* to the *business collaboration view*.

Having discussed which artifacts are maintained as extrinsic objects in the registry, we have to take care about facilitating the search for these objects. In other words, we have to define appropriate meta data that is attached to the extrinsic objects in the registry. The meta data must support typical search criteria that are used when querying for global and local choreographies. A candidate for such a criterion are the authorized roles, in order to select choreographies that are executed by a certain role. Another search criterion are the actions, i.e. the names of the steps in a business process, in order to find all choreographies that comprise a certain step. Furthermore, a definition is given for each UMM artifact in its corresponding tagged value. A substring search on these definitions is considered to be useful. In summary, we use the following three types of meta data in our registry approach:

- role
- action
- definition

All of these criteria are defined within the UML model of a choreography. However, as said before, a registry query does not address the content of the encapsulated extrinsic object. Consequently, the registry management has to extract roles, actions, and definitions from a submitted XMI file and it has to attach it as meta data to the extrinsic object of the XMI file.

An ebXML registry provides the concepts of classification schemes and slots for the purpose of managing meta data. The former is used if the instances used in the meta data follow an existing predefined classification. Since UN/CEFACT provides a list of authorized roles to be used in business collaborations and their transactions, the meta data for a role is best represented by a classification scheme. In the registry meta model in figure 7, we recognize a predefined classification schema for role. It consists of several classification nodes for the different pre-defined roles. Each classification node of a role (e.g. vendor) may be used in different choreographies. The classification of the so-called participating role refers to a usage of a role in a choreography. Evidently, a classification node of a role has multiple participating role classifications. Actions as well as definitions do not follow a pre-defined schema. Thus, they present subtypes of slots.

In the registry meta model in figure 7 the assignment of meta data to the extrinsic objects is denoted by *composition* and *aggregation* relationships. The black diamond of a composition denotes that a definition slot is exclusively assigned to an extrinsic objects. Evidently, the definition is made for one artifact and is not used for another one. The white diamond of an aggregation denotes that an action slot and a participating role classification is not exclusively assigned and, thus, may be shared between different extrinsic objects. The rational behind sharing these kinds of meta data is to establish a strong link between the actions and authorized roles in a global choreography and in a local one. In fact, the *local choreography view* describes a local process of an *authorized role* that matches exactly the *authorized role* of a *business collaboration protocol*. Similarly, the *initiating* and *reacting activities* in a local process match exactly the *requesting* and *responding actions* in a *business transaction*. Thus, we share the role classification between *local choreography view* and *business collaboration view* as well as the action slot between *local choreography view* and *business transaction view*.

In order to find potential business partners and to advertise oneself to other business partners, it is important that a business partner represents himself and his supported choreographies within the registry. For this purpose, we define *business partner* as a type of an ebRIM organization. In order to express the support of a global choreography, we establish a link from *business partner* to the *business collaboration view*. Evidently, the link from *business partner* to *local choreography view* expresses the support of a local choreography. These associations are of type *offers service*, which is pre-defined in ebRIM.

In addition, the meta model in figure 7 shows an *implements* association between the *local choreography view* and a *business service*, which in turn is linked to extrinsic objects for WSDL and BPEL files. This provides a short-cut for a Web Services definition and should just high-light where to interlink with corresponding Web Services artifacts, which is not detailed in this paper.

In section 2 we used our `order from quote` example to demonstrate global choreographies and in section 3 we exemplified a corresponding local choreography. Figure 8 shows how the artifacts introduced in these two sections are maintained and interlinked in our registry. The object diagram in figure 8 presents a valid instance of our registry meta model.

The registry example includes an extrinsic object for the full business collaboration model `order from quote`. It contains two ex-

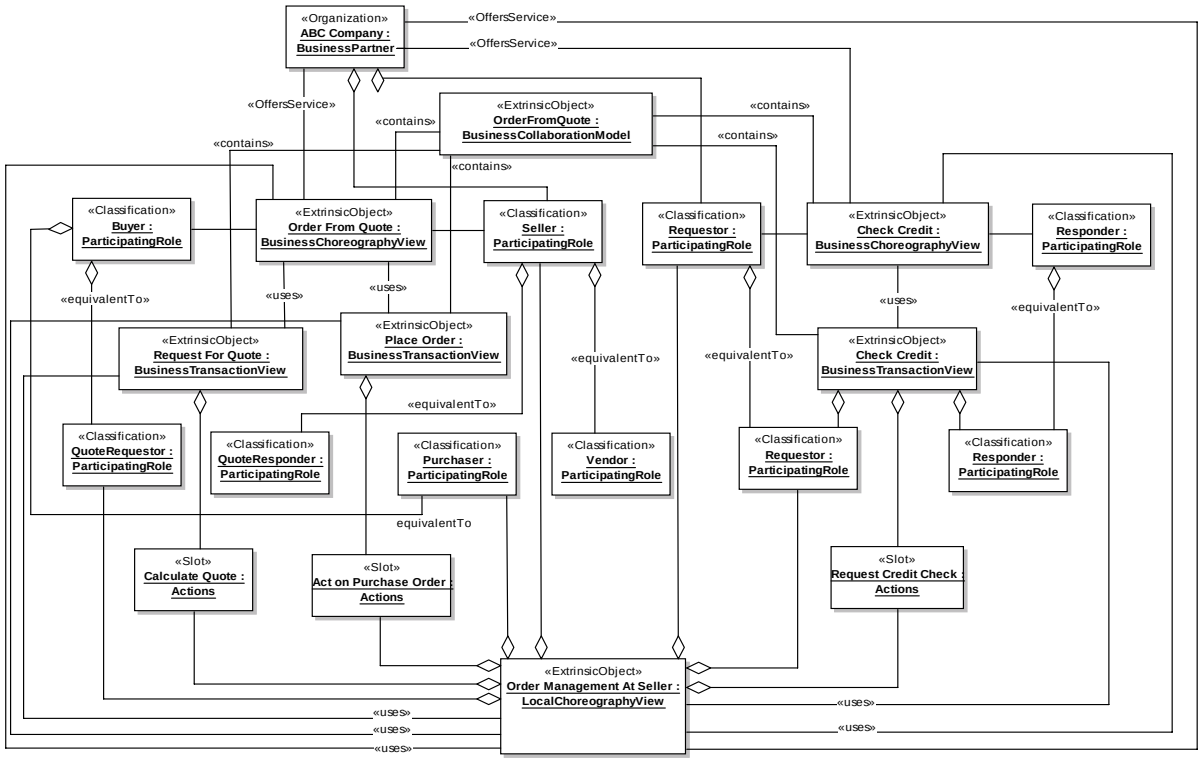

Figure 8: Example Registry Instance

trinsic objects of type *business choreography view* - `order from quote` and `check credit` - as well as another three of type *business transaction view*: `request for quote`, `place order`, and `check credit`. We add *uses* associations from the extrinsic objects of the *business choreography view* to the ones of the *business transaction view* in accordance to the UMM model. Another extrinsic object is the *local choreography view* `order management at seller`. In this example, we assume that it has been submitted separately from the full *business collaboration model* `order from quote`, since there is no association between them. Nevertheless, the *uses* associations from the *local choreography view* to the two *business choreography views* and to the three *business transaction views* have to be created by the registry management.

Furthermore, figure 8 shows some of the meta data associated to our example. We depict all participating role classifications for all views. Furthermore, we denote the logical mapping from roles in the *business collaboration view* to the roles in the *business transaction view*. For example, the `order from quote` *business choreography view* includes the roles `buyer` and `seller`, whereas the `request for quote` *business transaction view* includes the roles `quote requestor` and `quote responder`. By *equivalent to* associations we denote that the `buyer` plays the `quote requestor` and the `seller` the `quote responder` in the underlying view.

The example also visualizes the sharing of the participating roles classification between the *business choreography view* and the *local choreography view*. The participating role `buyer` of the `order from quote` *business choreography view* is exactly the same participating role as the one in the `order management at seller` *local choreography view*. Similarly, the sharing of actions is demonstrated. For example, the action `calculate quote` is part of the `request for quote` *business transaction view* as well as of the `order management at seller`, which realizes the former one. Please, note that we omit to show the actions of the seller and do not depict any definition slots in figure 8 for reasons of simplicity.

Finally, our example shows the ABC company being a business partner. Since it supports the shown global and local choreographies, the ABC company has *offers service* associations to the *business collaboration views* `order from quote` and `check credit` as well as to *order management at seller*. Furthermore, the meta data of the ABC company denotes that it is able to take on the `seller` (in `order from quote`) and the `requestor` (in `check credit`).

5. RELATED WORK

The idea of defining business processes crossing organizational boundaries goes back to ISO's Open-edi reference model [9]. A first implementation of the choreography aspects of this model was a Petri-Net approach contributed by Lee [15]. Also other authors used Petri-Nets to define the workflow between organizations [16],[18],[28]. In addition to the Petri-Nets formalism, conceptual modeling languages became popular for describing inter-organizational processes for the purpose of understanding and communication. The two most significant techniques for conceptual modeling of business processes are the Unified Modeling Language (UML) [2] and the Business Process Modeling Notation (BPMN) [32].

In order to use UML for modeling business processes different authors have developed either just guidelines or a UML profile that customizes UML for business process modeling. Customizations of UML for modeling business processes internal to a company are described in [12],[19],[25]. Beside UMM, UML customizations for

modeling inter-organizational processes are described in the RosettaNet Framework [13] and in Kramler et al. [14].

Another popular way of describing (inter-organizational) business processes was triggered by the growing importance of XML and Web Services. Different text based process modeling languages appeared. These usually had no graphical notation, but may be interpreted by software allowing the tracking or even execution of the business process. The Business Process Execution Language for Web Services (BPEL4WS) [23],[17] became the most popular language in this area. It is able to describe the orchestration of executable business processes, but also the message exchanges in a local choreography.

In the area of Web Services, the Web Services Choreography Description Language (WS-CDL) [10] is the choice for modeling global choreographies. However, WS-CDL uses its own set of control flow constructs which are hard to map to those of BPEL. In order to overcome this limitation, BPEL4chor [3] has recently been proposed to extend BPEL for describing global choreographies. Another XML-based language for describing global choreographies is ebXML's Business Process Specification Schema (BPSS) [22]. However, it should be noted that BPSS is based on UMM and is more or less an XML schema representing UMM's business collaboration protocol and business transaction [8].

Already in 2004 we investigated the interdependencies of UMM and BPEL in [7]. The subject of the paper was to scrutinize whether BPEL is appropriate for capturing the choreography modeled in UMM. However, going directly from UMM to BPEL does not allow to derive local choreographies between multiple parties. Furthermore, it cannot result in executable processes, since this approach does not cover all the internal activities in an orchestration. Another approach defining a direct mapping from a global choreography to a BPEL process is delivered by Khalaf [11]. He transforms the global choreography of a RosettaNet model into BPEL.

There exist some approaches that take care of the relationship between orchestration, local choreography, and global choreography [30] [29] [26].These approaches differ significantly from our proposed approach. Evidently, they are based on Petri Nets and Web Services, respectively, whereas our approach extends the UML. However more importantly, those approach deal with the automatic transformation between orchestration, local choreography, and global choreography and consistency checking between the different perspectives. Our approach provides a mechanism to specify the dependencies between global and local choreographies. These dependencies may be used in a reasoning approach to check consistency following the ideas of the other approaches.

Research in the field of registry models and registry classifications has brought a set of challenging approaches. In [20] an ontology driven registry classification model is presented. Using OWL [31] Liu et al. describe an ontology for an ebXML registry. Thereby they abstract from the original ebRIM specification by describing the registry classification using OWL in order to retrieve a registry classification model from it.

Another approach using OWL to extend ebRIM is presented in [33] and [4]. Roh et al. are using OWL to build a foundation for intelligent information processing in ebXML registries. Thereby they propose a new ebXML registry information model called `semantic information model` (SIM). Dogac et al. are presenting a mapping mechanism between the various constructs of OWL and the ebXML classification hierarchy.

None of these ebRIM extensions however is specifically dedicated to the storage and retrieval of global and local choreographies and to maintaining their interrelationships.

In the field of UDDI-based registries [21] (Universal Description, Discovery and Integration) [1] proposes an extension to the current UDDI standard called UDDIe. Using the concept of so called `blue pages` the UDDIe allows to store user defined properties associated with a specific service. This greatly enhances the process of service discovery and retrieval since additional meta-information can be stored in the registry.

6. SUMMARY

In this paper we presented an approach for registering global and local choreographies within an ebXML registry. Maintaining both kinds of choreographies within a registry helps business partners to engage in inter-organizational business processes. This is motivated by a top-down approach to establish business collaborations. It is expected that global choreographies for well-accepted business scenarios are developed by standard organizations, industry consortia, and market leaders. The global choreographies denoted by UMM serve as an agreement of the business process to be executed between business partners. Based on the commonly agreed UMM models, business partners may derive their local choreographies based on our dedicated UML profile and advertise it in the registry.

Registering global choreographies helps to attract new users in order to create a snowball effect to reach critical masses for certain business scenarios. Furthermore, its registration also fosters re-use, because parts of an available global choreography may be re-used by to-be-developed ones. Registering companies and associating them with global choreographies helps to find potential business partners that share a common business scenario in a complementary role. Once a business partner is found, the registry link to his local choreography provides information to binding to its services. This becomes even more powerful if the local choreography is associated with the corresponding BPEL and WSDL files, which is only highlighted, but not detailed in this paper. Furthermore, registered local choreographies may serve as real-world examples for business partners who want to implement the same role in a global choreography.

7. REFERENCES

[1] A. S. Ali, O. F. Rana, R. Al-Ali, and D. W. Walker. UDDIe: an extended registry for Web services. In *Symposium on Applications and the Internet*, 2003.

[2] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley, 2005. 2nd Edition.

[3] G. Decker, O. Kopp, F. Leymann, and M. Weske. Bpel4chor: Extending bpel for modeling choreographies. In *Proceedings of the 2007 IEEE International Conference on Web Services (ICWS 2007)*, pages 296–303. IEEE Computer Society, 2007.

[4] A. Dogac, Y. Kabak, and G. B. Laleci. Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery. In *14th International Workshop on Research Issues on Data Engineering*, 2004.

[5] M. Hammer and J. Champy. *Reengineering the Corporation:*

A Manifesto for Business Revolution. Harper Business, 1993.

[6] B. Hofreiter. Extending UN/CEFACTs modeling methodology by a UML profile for local choreographies. *Journal on Information Systems and E-Business Management*, 0(0), 2008. to appear, online first: 10.1007/s10257-008-0083-3.

[7] B. Hofreiter and C. Huemer. Transforming UMM Business Collaboration Models to BPEL. In *OTM 2004 Workshops*, volume 3292 of *LNCS*, pages 507–519, 2004.

[8] B. Hofreiter, C. Huemer, and J.-H. Kim. Choreography of ebxml business collaborations. *Inf. Syst. E-Business Management*, 4(3):221–243, 2006.

[9] ISO. Open-edi reference model, 1995. ISO/IEC JTC 1/SC30 ISO Standard 14662.

[10] N. Kavantzas, D. Burdett, G. Ritzinger, . T. Fletcher, Y. Lafon, and C. Barreto. Web services choreography description language, 2005. Version 1.0, http://www.w3.org/TR/ws-cdl-10/.

[11] R. Khalaf. From RosettaNet PIPs to BPEL processes: A three level approach for business protocols. *Data Knowl. Eng.*, 61(1):23–38, 2007.

[12] B. Korherr and B. List. Extending the uml 2 activity diagram with business process goals and performance measures and the mapping to bpel. In *Advances in Conceptual Modeling - Theory and Practice, ER 2006 Workshops*, volume 4231, pages 7–18. Springer LNCS, 2006.

[13] D. Kraemer and P. Yendluri. Realizing the benefits of implementing rosettanet implementation framework (rnif) version 2.0, 2002. http://www.rosettanet.org/cms/export/sites/default/RosettaNet/Downloads/whitePapers/RNIF2finalv3.pdf.

[14] G. Kramler, E. Kapsammer, G. Kappel, and W. Retschitzegger. Towards using uml 2 for modelling web service collaboration protocols. In *Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'05)*, 2005.

[15] R. M. Lee. Documentary petri nets: A modeling representation for electronic trade procedures. In *Business Process Management: Models, Techniques, and Empirical Studies*. Springer, 2000.

[16] K. Lenz and A. Oberweis. Inter-organizational business process management with xml nets. In *Petri Net Technology for Communication-Based Systems*, volume 2472, pages 243–263. Springer LNCS, 2003.

[17] F. Leymann, D. Roller, and M.-T. Schmidt. Web services and business process management. *IBM Systems Journal*, 41(2), 2002.

[18] S. Ling and S. W. Loke. Advanced petri nets for modelling mobile agent enabled interorganizational workflows. In *Proceedings of the 9th IEEE International Conference on Engineering of Computer-Based Systems (ECBS 2002)*, pages 245–252. IEEE Computer Society, 2002.

[19] B. List and B. Korherr. A uml 2 profile for business process modelling. In *ER 2005 Workshops Proceedings*, 2005.

[20] W. Liu, K. He, and W. Liu. Design and realization of ebXML registry classification model based on ontology. In *International Conference on Information Technology: Coding and Computing (ITCC)*, volume 2, pages 809–814, 2005.

[21] OASIS. *UDDI Specification 3.0.2*, Oct. 2004. 'http://uddi.org/pubs/uddi_v3.htm'.

[22] OASIS. *ebXML Business Process Specification Schema Technical Specification 2.0.4*, Dec. 2006. http://docs.oasis-open.org/ebxml-bp/2.0.4/ebxmlbp-v2.0.4-Spec-os-en.pdf.

[23] OASIS. *Web Services Business Process Execution Language Version 2.0*, Jan. 2007. Version 2.0, Commitee Specification.

[24] C. Peltz. Web services orchestration and choreography. *IEEE Computer*, 36(10):46–52, 2003.

[25] M. Penker and H.-E. Eriksson. *Business Modeling With UML: Business Patterns at Work*. Wiley, 2000.

[26] G. Piccinelli, W. Emmerich, C. Zirpins, and K. Schütt. Web service interfaces for inter-organisational business processes: An infrastructure for automated reconciliation. In *Proceedings of the 6th International Enterprise Distributed Object Computing Conference (EDOC 2002)*, pages 285–292. IEEE Computer Society, 2002.

[27] UN/CEFACT. *UN/CEFACT's Modeling Methodology (UMM), UMM Meta Model - Foundation Module*, Mar. 2006. Technical Specification V1.0, http://www.unece.org/cefact/umm/UMM_Foundation_Module.pdf.

[28] W. M. van der Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.

[29] W. M. P. van der Aalst. Inheritance of interorganizational workflows to enable business-to-business. *Electronic Commerce Research*, 2(3):195–231, 2002.

[30] W. M. P. van der Aalst and M. Weske. The p2p approach to interorganizational workflows. In *Advanced Information Systems Engineering, 13th International Conference, CAiSE 2001*, volume 2068, pages 140–156. Springer LNCS, 2001.

[31] W3C. *Web Ontoloty Language (OWL)*. W3C. http://www.w3.org/2004/OWL/.

[32] S. A. White. Business process modeling notation specification 1.0, 2006. http://www.omg.org/docs/dtc/06-02-01.pdf.

[33] R. Yohan, K. Hangkyu, K. H. Soo, K. M. Ho, and S. J. Hyun. Semantic business registry information model. In *International Conference on Convergence Information Technology*, 2007.