UN/CEFACT'S Modeling Methodology (UMM): A UML Profile for B2B e-Commerce

B. Hofreiter[1], C. Huemer[1,3], P. Liegl[2], R. Schuster[2], and M. Zapletal[3]

[1] University of Vienna
{birgit.hofreiter, christian.huemer}@univie.ac.at
[2] Research Studios Austria DME
{pliegl, rschuster}@researchstudio.at
[3] Vienna University of Technology
marco@ec.tuwien.ac.at

Abstract. The United Nation's Centre for Trade Facilitation and Electronic Business (UN/CEFACT) is an e-business standardization body. It is known from its work on UN/EDIFACT and ebXML. One of its on-going work items is the UN/CEFACT modeling methodology (UMM) for modeling global choreographies of B2B scenarios. The goal of UMM is defining a shared business logic between business partners and fostering reuse of standardized process building blocks. The latest UMM version is defined as a UML 1.4 profile. In this paper we introduce the main concepts of UMM to realize its vision. Furthermore, the paper elaborates on the necessary UML meta model work-arounds we - as part of the specification's editing team - took in order to accomplish the B2B requirements. Then we propose a move towards UML 2 that eliminates some of those workarounds.

1 Introduction

Automating the exchange of business information between business partners exists for a while. In the early days of electronic data interchange (EDI) the focus was limited to standardizing the business document types. However, the business documents must also be exchanged in an agreed order. The business processes between two different organizations participating in a collaborative business process must be defined. For this purpose a commonly accepted methodology is needed.

The United Nation's Centre for Trade Facilitation and Electronic Business (UN/CEFACT), known for its standardization work in the field of UN/EDIFACT and ebXML, took up the endeavor and started research for such a methodology. This on-going work resulted in UN/CEFACT's Modeling Methodology (UMM). UMM enables to capture business knowledge independent of the underlying implementation technology, like Web Services or ebXML. The goal is to specify a global choreography of a business collaboration serving as an "agreement" between the participating business partners in the respective collaboration. Each business partner derives in turn its local choreography, enabling the configuration of the business partner's system.

J.F. Roddick et al. (Eds.): ER Workshops 2006, LNCS 4231, pp. 19–31, 2006.

In order to guarantee user acceptance of the UMM, it must be both effective and easy to understand for the business process modelers and software architects. Due to the growing tool support of the Unified Modeling Language (UML), the decision in favor of UML as notation of UMM was already made in 1998. In the first years, UMM specified its own conceptual meta model and provided guidelines on creating compliant artifacts using the UML. In late 2004 it was decided to define the most recent UMM version as a UML profile [1], i.e. a set of stereotypes, tagged values and constraints - in order to customize the UML meta model for the special purpose of modeling global B2B choreographies. At this time the UML version of choice by UN/CEFACT was UML 1.4 [2]. This paper introduces the most important concepts of the UML 1.4 profile of UMM. Most attention is spent on necessary work-arounds in order to adjust the UML meta model to the special needs of UMM. A future transition of UML 1.4 to UML 2 as the basis of UMM will affect its UML profile. Thus, we will highlight the potential of such a move forward.

2 Related Work

In the world of Web Services a lot of different languages describing business processes exist, e.g. the Business Process Modeling Language (BPML) [3] and the Business Process Execution Language (BPEL) [4]. These languages are limited to orchestrations and local choreographies. The release of the Web Services Choreography Description Language (WS-CDL) draft [5] adds a specification for global choreographies to the family of Web Services which did not exist before. Within the ebXML framework, the Business Process Specification Schema (BPSS) [6] always describes the choreography of a business collaboration from a global perspective.

Since all the above mentioned languages are XML-based, there have been attempts to model them in a graphical syntax and/or to apply a model driven approach leading to them. The model driven approach is also in-line with the Open-edi reference model that became an ISO standard in 1997 [7]. 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. Accordingly, special UML profiles may be used on the BOV level, whereas the Web Services and ebXML languages are on the FSV level.

Several approaches using UML for business process modeling have been proposed [8] [9] [10]. However, these approaches focus on the modeling of business processes internal to an organization. Other approaches use UML to visualize Web Services and their choreography [11] [12]. More advanced approaches provide a development process for inter-organizational business processes. These are either driven by existing private workflows [13] or they are driven by the inter-organizational requirements instead of the private ones [14].

Also the UMM, which presents the core of this paper, is considered as a BOV-centric methodology. When UN/CEFACT and OASIS started the ebXML inititative, it was UN/CEFACT's vision that UMM is used to create BOV standards and that XML is used as key concept on the FSV layer. Accordingly, UMM is ebXML's modeling methodology, but it is not a mandatory part of ebXML (c.f. [6]). Since UMM stops at the BOV layer, a transformation to an IT solution on the FSV layer is required. In [15] we describe such a mapping from UMM to BPEL. Furthermore, we define a mapping from UMM models to BPSS in [16].

3 UMM by Example

In this section we briefly describe the steps of UMM and the resulting artifacts. For a better understanding we walk through the UMM by the means of a rather simple, but still realistic example. This example is akin to a project in the European `waste management` domain. Crossborder transports of waste - even within the EU - are subject to regulations. A transport must be announced, the receipt of the waste as well as the disposal of the waste must be signaled. Exporter, importer, the competent authorities in their countries and in transit countries interchange this information. In order to keep the example simple we do not consider the competent authorities of transit and we do not include the information about the waste disposal. However, in order to explain all concepts we assume that each individual transport must be approved which is not required in reality.

The UMM comprises three main views: *business domain view* (BDV), *business requirements view* (BRV), and *business transaction view* (BTV). The latter two are split into subviews. A UMM *business collaboration model* reflects this structure by creating packages for all these views and subviews (See left hand side of figure 1).

The BDV is used to gather existing knowledge from stakeholders and business domain experts. In interviews the business process analyst tries to get a basic understanding of the business processes in the domain. The use case descriptions of a *business process* are on a rather high level. One or more *business partners participate* in a *business process* and zero or more *stakeholders* have an *interest in* dependency with the process. The BDV results in a map of *business processes*, i.e. the *business processes* are classified. Thus the BDV package includes *business area* subpackages. UN/CEFACT suggests to use *business areas* according to the classification of Porters value chain (PVC) plus some administrative areas. Each *business area* consists of *process area* packages that correspond to the Open-edi phases (planning, identification, negotiation, actualization, and post-actualization) [7]. In our `waste management` example relevant *business areas* are logistics and regulation, each covering at least the *process areas* of actualization and post-actualization. We do not want to detail here all the processes that may be important to the domain experts and stakeholders in these areas.

Those *business processes* from the BDV that provide a chance for collaboration will be further detailed by the business process analyst in the BRV. The BRV consists of a number of different subviews. The *business process view*

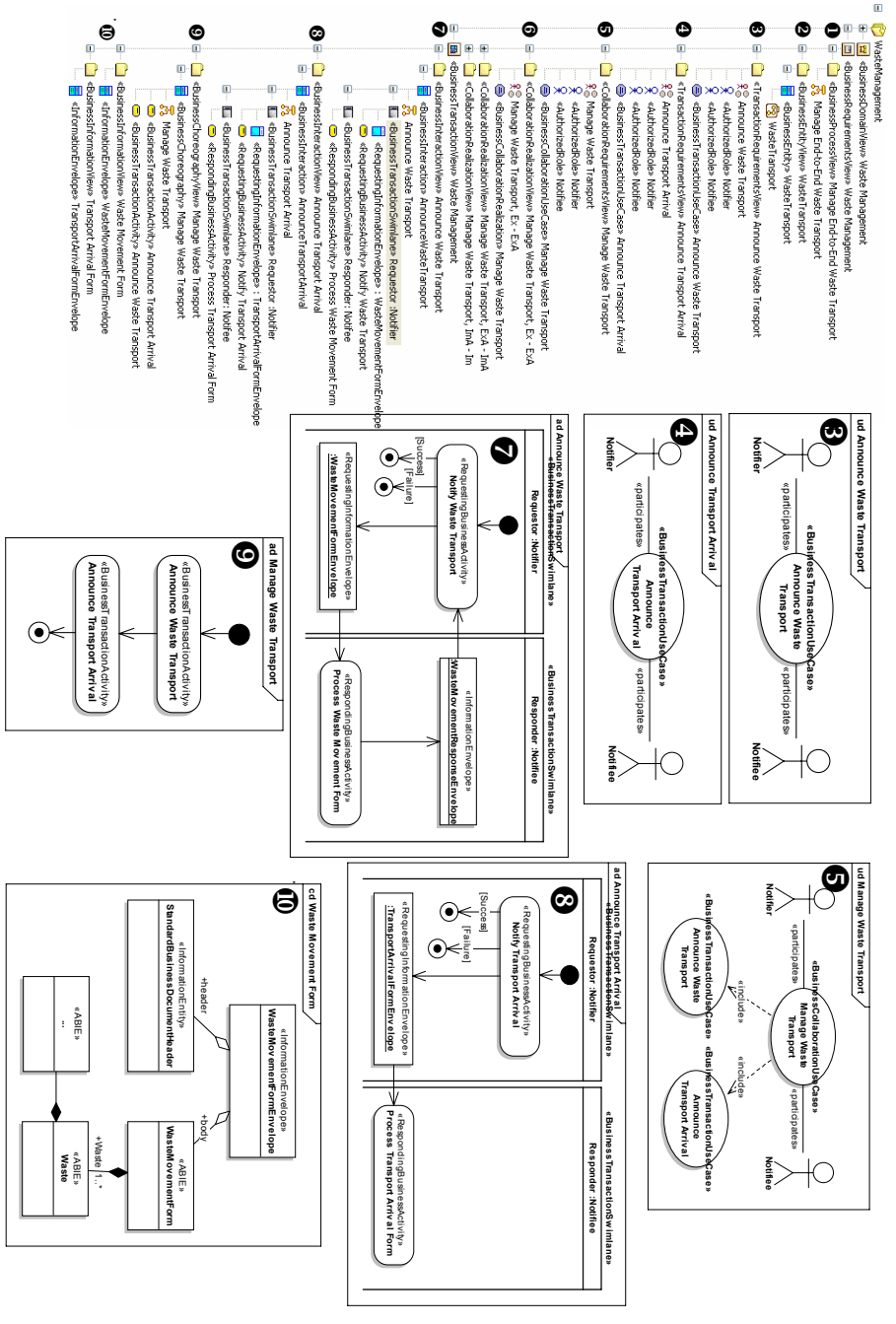

Fig. 1. UMM Overview

(1 in figure 1) gives an overview about the *business processes*, their activities and resulting effects, and the *business partners* executing them. The activity graph of a *business process* may describe a single partner's process, but may also detail a multi-party choreography. The business process analyst tries to discover interface tasks creating/changing *business entities* that are shared between *business partners* and, thus, require communication with a *business partner*. In our example we detail a multi-party *business process* for a `waste transport`. The `exporter` pre-informs the `export authority` about a waste transport and expects an approval of the waste transport in return. In turn, the `export authority` announces the waste transport to the `import authority` to get the approval, and the `import authority` does the same with the `importer`. Later on when the waste is received by the `importer`, this information goes uni-directional back the chain from the `importer` to the `import authority` to the `export authority`, and finally to the `exporter`.

The information exchanged between *business partners* is about the *business entity* `waste transport`. Firstly, a `waste transport` entity is created with state `announced`. `Announced` is a kind of pending state because it requires a decision by the other *business partner* to set it either to `approved` or to `rejected`. Once an approved transport has happened it is set to `arrived`. These so-called *shared business entity states* must be in accordance with the *business entity lifecycle* of `waste transport`. This lifecycle is defined in the state chart of the *business entity view* (2).

It is obvious from the requirements described so far that the announcement together with the information of approval/rejection as well as the information of the waste receipt always occur between a different pair of *business partners*. It is not efficient to describe these tasks for each pair again and again. Instead, these tasks are defined between *authorized roles*. A *transaction requirements view* defines the *business transaction use case* for a certain task and binds the two *authorized roles* involved. In our example we have two *transaction requirement views*: `announce waste transport` (3) - which also includes the decision - and `announce transport arrival` (4). The *authorized roles* are in both cases a `notifier` who makes the corresponding announcement and a `notifee`.

The *collaboration requirements view* includes a *business collaboration use case*. The *business collaboration use case* aggregates *business transaction use cases* and/or nested *business collaboration use cases*. This is manifested by *include* associations. In our example the *business collaboration use case* `manage waste transport` (5) includes the *business transaction use cases* `announce waste transport` (3) and `announce transport arrival` (4). Furthermore, the *authorized roles* participating in the *business collaboration use case* must be defined. Sometimes it is hard to find a good name for an *authorized role*, like in our example. We call the roles again `notifier` and `notifiee`. The `notifier` is the one who initiates the management of a waste transport and the `notifiee` is the one who reacts on it. A *business collaboration use case* may have many *business collaboration realizations* that define which *business partners* play which *authorized roles*. A detailed discussion of *business collaboration realizations* is provided in section 4.

The BTV builds upon the BRV and defines a global choreography of information exchanges and the document structure of these exchanges. The choreography described in the requirements of a *business transaction use case* is represented in exactly one activity graph of a *business transaction*. A *business transaction* is used to align the states of *business entities* in the information systems of the *authorized roles*. We distinguish one-way and two-way *business transactions*: In the former case, the initiating *authorized role* reports an already effective and irreversible state change that the reacting *authorized role* has to accept. This is the case in the *business transaction* `announce transport arrival` (8). In the other case, the initiating partner sets the *business entity/ies* into an interim state and the final state is decided by the reacting *authorized role*. It is a two-way transaction, because an *information envelope* flows from the initiator to the responder to set the interim state and backwards to set the final and irreversible state change. In the *business transaction* `announce waste transport` (7) the *business entity* `announce waste transport` is set into the interim state `announce` by the `notifer`, whereas the `notifee` sets the final state of `approved` or `rejected`. Irreversible means that returning to an original state requires compensation by another *business transaction*.

A UMM *business transaction* follows always the same pattern: A *business transaction* is performed between two *authorized roles* that are already known from the *business transaction use case* and that are assigned to exactly one swimlane each. Each *authorized role* performs exactly one activity. An object flow between the *requesting* and the *responding business activity* is mandatory. An object flow in the reverse direction is optional. Both the two-way transaction `announce waste transport` (7) and the one-way transaction `announce transport arrival` (8) follow this pattern.

The activity graph of a *business transaction* shows only the exchange of business information in the corresponding envelopes. It does not show any business signals for acknowledgements. The *acknowledgment of receipt* - sent for a valid document that also passed sequence validation - and the *acknowledgment of processing* - sent for documents that have been checked against additional business rules before importing them into the business application - are specified by maximum time values in the tagged values of the *requesting* and *responding business activity*. Further tagged values are the maximum *time to respond*, and flags for *authorization*, *non-repudiation* of original and of receipt, and a *retry counter* for reinitiating the *business transactions* in case of control failures. The *information envelopes* are characterized by tagged values to signal *confidential*, *tamper proof* and *authenticated* exchanges.

According to the UMM *business transaction* semantics, the *requesting business activity* does not end after sending the envelope - it is still alive. The *responding business activity* may output the response which is returned to the still living *requesting business activity*. This interpretation may be curious for UML purists - however, it was already introduced by the RosettaNet [17] modeling approach and is well accepted by the e-business community.

The requirements described in a *business collaboration use case* are choreographed in the activity graph of a *business collaboration protocol* which is defined in a *business choreography view*. In our example, the `manage waste transport` requirements (5) are mapped to the homonymous *business collaboration protocol* (9). A *business collaboration protocol* choreographs a set of *business transaction activities* and/or *business collaboration activities*. A *business transaction activity* is refined by the activity graph of a *business transaction*. In our example, the *business collaboration protocol* of `manage waste transport` (9) is a simple sequence of two *business transaction activities*: `announce waste transport` and `announce transport arrival`. Each of them is refined by its own *business transaction* (7,8). *Business transaction activities* have tagged values for a maximum *time to perform* and an indicator whether *concurrent* execution is allowed or not. *Business collaboration activities* - which are not used in our example - are refined by a nested *business collaboration protocol*.

Finally, the information exchanged in transactions must be unambiguously defined. Each object in an object flow state is an instance of a class representing an envelope. The aggregates within this envelope are defined in a class diagram. Figure 1 includes a - due to space limitations - very limited extract of the class diagram for the `waste movement form` envelope (10), which is exchanged in the *business transaction* `announce waste transport` (7). The business document is assembled from reusable building blocks called *core components* [18] [19]. By using a core component in a business document it is adjusted to the document's business context, e.g. by eliminating attributes not needed. Once the core component is adjusted it becomes a so-called *business information entity*. In (10) we just highlight one *business information entity* `waste` being part of a `waste movement form`. We do not list any attributes, neither we show any other *business information entities* and relationships among them.

4 UMM Meta Model Workarounds

As mentioned before, UMM is based on the UML 1.4 meta model [2]. Accordingly, a UMM *business collaboration model* is a UMM 1.4 compliant model. However, some concepts may appear unfamiliar to a UML modeler who has not used UMM before. These concepts are a result of the specific B2B requirements of UMM.

4.1 Mapping of Authorized Roles

One of the key goals of UMM is to foster reuse. This implies that a *business transaction use case* may be included in many *business collaboration use cases*. Consider for example, the *business transaction use case* `announce transport arrival` is part of another *business collaboration use case* in the logistics domain.

For the purpose of reuse, the *authorized roles* are defined in the very specific context of a *business transaction*. A *business collaboration use case* that includes the *business transaction* also defines participating *authorized roles* in its specific context. It is the peculiarity of UMM that a certain *authorized role* of the *business*

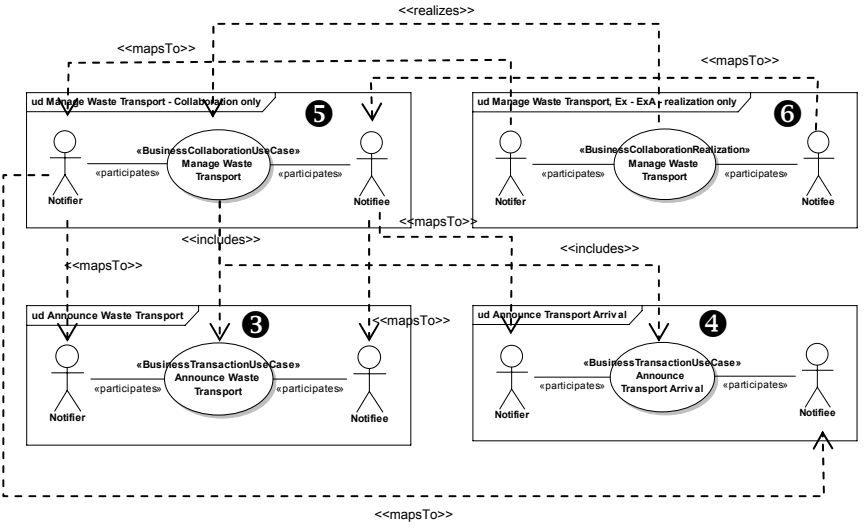

Fig. 2. Mapping of Authorized Roles

collaboration use case must take on an *authorized role* of an included *business transaction use case*. For this purpose UMM uses *maps to* dependencies defining which *authorized role* of a *business collaboration use case* plays which role in an included *business transaction use case* (or nested *business collaboration use case*).

This concept is easily demonstrated by our `waste management` example (see figure 1). The *business collaboration use case* `manage waste transport` (5) includes two *business transaction use cases*: `announce waste transport` (3) and `announce transport arrival` (4). By coincidence, the roles of all three use cases are `notifier` and `notifee`. However, this does not mean that a `notifier` always maps to a `notifier`. In our example the `notifier` of `manage waste transport` (5) plays also the `notifier` of `announce waste transport` (3), but plays the `notifiee` in `announce transport arrival` (4) since the information flows the other way round. For the `notifiee` of `manage waste transport` it is just the opposite. It is obvious, that `notifier` must be a different *authorized role* in each of the three use cases, however with a homonymous name. Accordingly, *authorized roles* are always defined in the namespace of its *transaction requirements view*. This is easy to recognize in the tree view of our example on the left side of figure 1.

In section 3 we already learned that the same `manage waste transport` *business collaboration* must be realized between different pairs of *business partners*. Thus, we need different *business collaboration realizations* - each defining which *business partner* plays which role in it. Accordingly, our waste example results in three *business collaboration realizations* of `manage waste transport` - one between `exporter` and `export authority`, one between `export authority` and `import authority`, and one between `import authority` and `importer`. Figure 2 depicts the first one as a representative (6). The *business partners* participating in the

business collaboration realization are the ones already defined in the BDV and, thus, are not re-defined in the namespace of the *collaboration realization view*. However, each *business collaboration realization* defines *authorized roles* which are usually - but not necessarily - homonymously named as the ones of the corresponding *business collaboration use case*. The previously introduced concept of *maps to* dependencies is used to map both the *authorized roles* from a *business collaboration realization* to a *business collaboration use case* as well as *business partners* to *authorized roles* of the *business collaboration realization*. In the `manage waste transport` realization (6) of figure 2 the `exporter` plays the `notifier` and the `export authority` acts as `notifiee`.

4.2 Reusing a Business Transaction in Many Business Collaboration Protocols

The fact that a *business transaction use case* (and a nested *business collaboration use case*) may be included in many *business collaboration use cases* has another implication which is not perfectly met by the UML 1.4 meta model. Each *business collaboration use case* leads to exactly one *business collaboration protocol*. Each included *business transaction use case* will result in a *business transaction* that is part of the corresponding *business collaboration protocol*. In the activity graph of a *business collaboration protocol* the activity graph of a *business transaction* is represented by a *business transaction activity*. Since a *business transaction use case* may be the target of many *include* associations, it follows that the same *business transaction* may be part of different *business collaboration protocols*.

One might think that this concept is reflected in the UML 1.4 meta model. A *business transaction activity* is a UML subactivity state which is refined by a UML activity graph. The UML 1.4 meta model defines a 1:1-relationship between subactivity states and activity graphs. However the relationship between *business transaction activity* and *business transaction* must be n:1. Consequently, UMM uses again a *maps to* dependency to realize the relationship between *business transaction activities* and *business transactions*. A *business transaction activity* is the source for only one *maps to* dependency and a *business transaction* may be the target of many *maps to* dependencies.

4.3 Mapping of Use Cases and Their Activity Graphs in Different Packages

Back in the early days of UMM development, in the late 1990s, the project team decided that the structure of a UMM model follows the UMM views (cf. left hand side of figure 1). In the meantime the user community got used to this structure and changing it would create too much confusion. In UML, an activity graph may go beneath a use case describing its requirements. However, in the UMM structure the *business transaction use cases* and the *business collaboration use cases* are located in other packages than the corresponding *business transaction* or *business collaboration protocol*, respectively. Accordingly, the relationship between a *business transaction use case* and a *business transaction* as well as

the one between a *business collaboration use case* and a *business collaboration protocol* are realized by another *maps to* dependency.

5 Moving Towards UML 2

With the growing acceptance of UML 2, there is a desire to move UMM towards UML 2. As we learned, UMM is based on use cases to model requirements, on classes to model business documents, and on activity diagrams to model choreographies. Since UML 2 made major changes to modeling activity diagrams, major changes must be made to UMM. In UML 1.4 *activity graphs* were specialized *state machines*. In UML 2 they have been replaced by the concept of an *activity*. An *activity* captures user-defined behavior by describing a flow of *actions* and their interaction with *objects* representing data. An *action* is a fundamental unit to describe a step of work within the execution of an *activity*. In this section we propose a transition of UMM *business collaboration protocols* and *business transactions* to UML 2 using the `waste management` example. Furthermore, we will show how to eliminate the workarounds introduced in section 4 using UML 2 concepts.

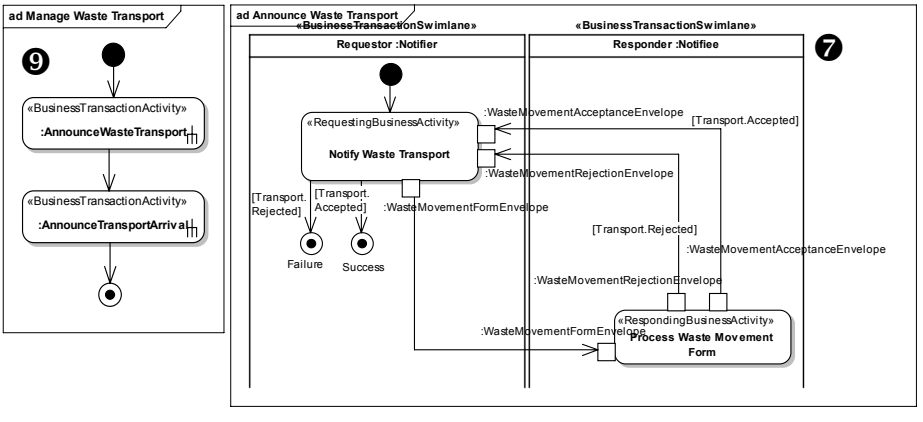

Fig. 3. Waste management example using UML 2

The *business collaboration protocol* `manage waste transport` on the left hand side of figure 3 is composed of two *business transaction activities* - `announce waste transport` and `announce transport arrival`. In UML 2 the *business collaboration protocol* becomes an *activity*. Each of the two *business transaction activities* become an *action*. We already know that *business transaction activities* are refined by a *business transaction*. A *business transaction* also becomes an *activity* in UML 2. In order to refer from a *business transaction activity* to its refining *business transaction* we utilize the predefined action type *call behavior action*. A *call behavior action* - indicated by the rake-symbol in the

lower right corner of a *business transaction activity* in figure 3 - allows the call of another *activity*. This eliminates the corresponding *maps to* dependency in the current UMM. In our example the first *business transaction activity* calls the `announce waste transport` *business transaction* and the second one calls the `announce transport arrival` *business transaction*.

A *business transaction* is always composed of a *requesting* and a *responding business activity* - each of them become *actions*. Since the implementation of these activities together with their interfaces within an application are partner specific we use the subtype *opaque action*. This indicates that the "semantics of the action are determined by the implementation" [20]. In UML 2 we prefer to notate the information flows between these two actions by the new pin notation. The right hand side of Figure 3 shows this new notation for the *business transaction* `announce waste transport`. An output pin of a *requesting business activity* and an input pin of a *responding business activity* are assigned with a *requesting information envelope* object. An output pin of a *responding business activity* and an input pin of a *requesting business activity* are assigned with a *responding information envelope* object.

Considering the response in case of two-way transactions, we suggest an extension to the UMM transaction concept. The current UMM transaction concept allows only one type of *responding information envelope*. Usually, the type of response differs significantly in case of a positive and a negative response. In the current UMM we must use an abstract super type for the positive and negative response. We propose multiple output pins to *responding business activities* and multiple input pins to *requesting business activities* in order to show different types of object flows. These object flows are guarded by mutually exclusive constraints. The `announce waste transport` *business transaction* on the right hand side of figure 3 defines the exchange of a `waste movement form envelope` between the *requesting business activity* `notify waste transport` and the *responding business activity* `process waste movement form`. A `waste movement acceptance envelope` is returned in case of an accepted transport. A `waste movement rejection envelope` is sent back if the transport is rejected. This approach narrows the gap between process- and data modeling in UMM.

Last, but not least UML 2 enables eliminating the *maps to* dependency between a *business transaction* and a *business transaction use case* and also between a *business collaboration protocol* and a *business collaboration use case*. In UML 2, a use case might be associated to an arbitrary *classifier* indicating that the *classifer* realizes the use case. Since *activity* inherits from *classifier*, we are able to connect the activities to the corresponding use cases without *maps to* dependencies.

6 Summary

In this paper we have introduced UN/CEFACT's Modeling Methodology (UMM) which we have co-edited. UMM defines a UML 1.4 profile - i.e. a set of stereotypes, tagged values and constraints - in order to customize the UML meta

model for the special purpose of modeling collaborative business processes from a global view. We demonstrated the steps of the UMM by a simple example of the `waste management` domain. This example reveals most of the stereotypes defined in UMM. Due to space limitation we were not able to go into all the details of the tagged values of each stereotype. Furthermore, we preferred to show the relationships between the stereotypes by means of the example, rather than introducing the equivalent set of OCL constraints as defined in the UMM specification.

Furthermore, we elaborated those concepts that are very specific to UMM's UML profile. These specifics include the mapping of authorized roles participating in a parent use case to the authorized roles participating in an included use case, as well as on the mapping of business partners to authorized roles in a use case realization. Another UMM specialty is that an activity graph may be included in multiple parent activity graphs. Since the UML 1.4 meta model defines a 1:1 relationship between a subactivity state and the refining activity graph, we had to implement a workaround with dependencies between the subactivity state and the refining activity graph to realize an n:1-relationship. Finally, we also used dependencies to trace between a use case and its realizing activity graph located in different packages.

Due to the growing acceptance of UML 2, it is predictable that the UML profile will move towards UML 2 in the near future. Since the concept of modeling activity diagrams changed dramatically in UML 2, we outlined the consequences of such a movement. Furthermore, we have shown the significance of a UMM tool, supporting the modeler in creating a UMM compliant model. The University of Vienna and the Research Studios Austria are committed to the development of the UMM Add-In and will adapt it to future versions of the UMM standard.

References

1. UN/CEFACT Techniques and Methodologies Group: UN/CEFACT's Modeling Methodology (UMM), UMM Meta Model - Foundation Module. (2006) Candidate for 1.0, Final Working Draft, http://www.unece.org/cefact/umm/UMM_Foundation_Module.pdf.
2. Object Management Group (OMG): Unified Modeling Language Specification. (2005) Version 1.4.2, http://www.omg.org/docs/formal/05-04-01.pdf.
3. Arkin, A.: Business Process Modeling Language (BPML). Technical report (2002)
4. BEA, IBM, Microsoft, SAP AG and Siebel Systems: Business Process Execution Language for Web Services. (2003) Version 1.1, ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf.
5. World Wide Web Consortium (W3C): Web Services Choreography Description Language. (2005) Version 1.0, http://www.w3.org/TR/ws-cdl-10/.
6. UN/CEFACT Techniques and Methodologies Group: UN/CEFACT - ebXML Business Process Specification Schema. (2003) Version 1.10, http://www.untmg.org/dmdocuments/BPSS_v110_2003_10_18.pdf.
7. ISO: Open-edi Reference Model. (1995) ISO/IEC JTC 1/SC30 ISO Standard 14662.

8. Penker, M., Penker, M., Eriksson, H.E.: Business Modeling With UML: Business Patterns at Work. Wiley (2000)
9. Vasconcelos, A., Caetano, A., Neves, J., Sinogas, P., Mendes, R., Tribolet, J.: A framework for modeling strategy, business processes and information systems. In: EDOC '01: Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing, IEEE Computer Society (2001)
10. List, B., Korherr, B.: A uml 2 profile for business process modelling. In: ER 2005 Workshops Proceedings. (2005)
11. Gardner, T.: UML Modelling of Automated Business Processes with a Mapping to BPEL4WS. In: 1st European Workshop on Object Orientation and Web Services (EOOWS'03), Springer (2003)
12. Thöne, S., Depke, R., Engels, G.: Process-oriented, flexible composition of web services with uml. In: Conceptual Modeling - ER 2002, 21st International Conference on Conceptual Modeling, Proceedings. LNCS, Springer (2002)
13. Jung, J.Y., Hur, W., Kang, S.H., Kim, H.: Business process choreography for b2b collaboration. IEEE Internet Computing **8**(1) (2004) 37–45
14. Kramler, G., Kapsammer, E., Kappel, G., Retschitzegger, W.: 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. Hofreiter, B., Huemer, C.: Transforming UMM Business Collaboration Models to BPEL. In: Proceedings of OTM Workshops 2004. Volume 3292., Springer LNCS (2004) 507–519
16. Hofreiter, B., Huemer, C., Kim, J.H.: Choreography of ebXML business collaborations. Information Systems and e-Business Management (ISeB) (2006) Springer.
17. RosettaNet: RosettaNet Implementation Framework: Core Specification. (2002) V02.00.01, http://www.rosettanet.org/rnif.
18. UN/CEFACT Techniques and Methodologies Group: Core Components Technical Specification - Part 8 of the ebXML Framework. (2003) Version 2.01, http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf.
19. UN/CEFACT Techniques and Methodologies Group: UML Profile for Core Components based on CCTS 2.01. (2006) Candidate for Version 1.0.
20. Object Management Group (OMG): Unified Modeling Language Specification. (2007) Version 2.0, http://www.omg.org/docs/formal/05-07-04.pdf.