UN/CEFACT's Modeling Methodology (UMM) in a Nutshell

UN/CEFACT TMG, Contact: Christian Huemer, huemer@big.tuwien.ac.at

UN/CEFACTs Modeling Methodology (UMM) is a UML modeling approach to design the business services that each business partner must provide in order to collaborate. It provides the business justification for the service to be implemented in a service-oriented architecture (SOA). In this paper 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 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. The *business partner types* notifer, consignee, 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 relevant artifacts of our example are depicted in the figure. On the left hand side of this figure we see the structure of our `waste management` model. A UMM *business collaboration model* comprises three main views: the *business domain view* (BDV), the *business requirements view* (BRV), and the *business transaction view* (BTV). The three top level packages of any UMM model - each highlighted with a star in the structure of the figure - are always stereotyped in accordance to these views.

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 partner types 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 postactualization). 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 *business processes* that may be important to the domain experts and stakeholders in these areas.

The BRV consists of a number of different subviews. The *business process view* (1) and the *business entity view* (2) are both very project specific. The *business process view* 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 partners 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. Discovery of interface task is more important in this step then modeling an exact control flow of activities. In our example we detail a multi-party business process for a `waste transport`. The `notifer` pre-informs the `export authority` about a waste transport. The `export authority` then pre-informs the `import authority`, and the `import authority` pre-informs the `consignee`. The approval of the waste transport goes the opposite direction along the chain. Once the waste is received the information goes backward the chain as well.

The information exchanged between *business partners* is about the *business entity* `waste transport`. Firstly, a `waste transport` object is created with state `announced`. Later it is set to the state `approved` and finally to the state `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 easy to recognize from the requirements captured so far that the same tasks always take place between a different pair of *business partner types*. Thus, it is not appropriate 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. The *authorized roles* are defined in the exact context of the *business transaction use case*. In our example we have two *transaction requirement views*: `announce waste transport` (3), which spans over the announcement and the approval, and `announce transport arrival` (4). The *authorized roles* are in both cases an `information provider` who makes the corresponding announcement and an `information consumer`. However, the *authorized role* `information provider` in `announce waste transport` is not the same as the one in `announce transport arrival`. This means, we have two *authorized roles* `information provider`, each defined in the namespace of its *transaction requirements view*. On the left hand side of the figure we see that both *transaction requirements views* (3,4) include an `information provider`. Of course the same is valid for the `information consumer`.

The *collaboration requirements view* includes a *business collaboration use case*. The *business collaboration use case* aggregates *business transaction use cases* or nested *business collaboration use cases*. This is manifested by *in-*

clude 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 within the context and namespace of the *collaboration requirements view*. We call the roles `outbound role` and `inbound role`. The `outbound role` is the one who initiates the management of a waste transport and the `inbound role` is the one who reacts on it. *Maps to* dependencies are used to define 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*). In our example the `outbound role` of `manage waste transport` (5) plays the `information provider` of `announce waste transport` (3), but plays the `information consumer` in `announce transport arrival` (4) since the information flows the other way round. For the `inbound role` of `manage waste transport` it is just the opposite.

A *collaboration realization view* covers a *business collaboration realization* - which is a kind of use case that does not elaborate any new requirements. A *business collaboration realization* realizes a *business collaboration use case* between a specific set of *business partner types*. This is indicated by a *realize* association. The *business collaboration realization* `manage waste transport` (6a) realizes the *business collaboration use case* with the same name (5). The *business partner types* participating in the *business collaboration realization* are the ones already defined in the BDV and, thus, are not redefined in the namespace of the *collaboration realization view*. A *maps to* dependency defines which participant of a *business collaboration realization* plays which role of the *business collaboration use case*. In the first `manage waste transport` realization (6a) the `notifer` plays the `outbound role` and the `export authority` acts as `inbound role`. The two additional `manage waste transport` realizations between `export authority` and `import authority` and between `import authority` and `consignee` are not depicted in the figure. This concludes all the subpackages of the BRV.

The BTV builds upon the BRV and defines a global choreography of information exchanges and the document structure of these exchanges. In this paper we concentrate on the choreography aspect and do not consider the document structures any further.

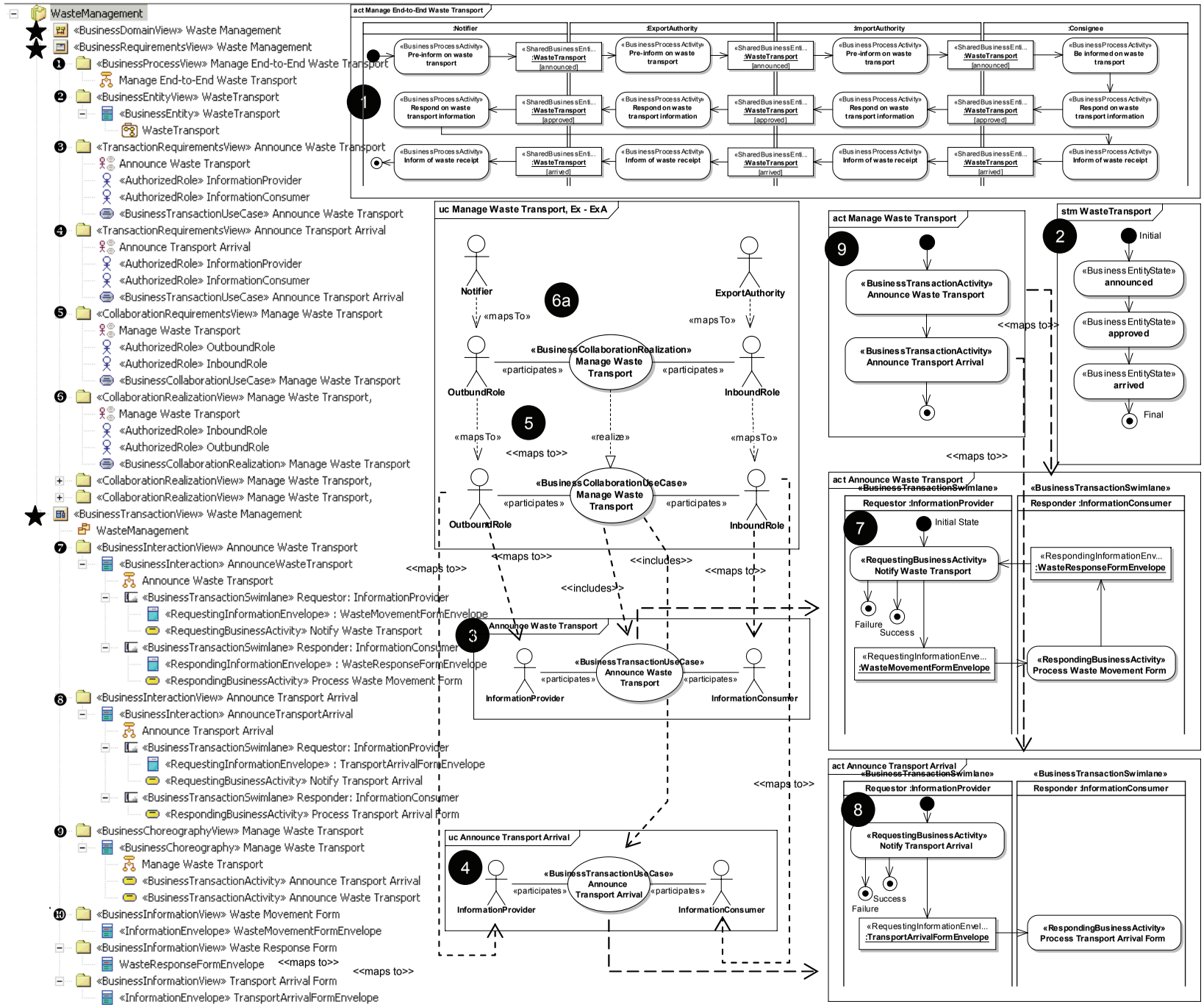
The choreography described in the requirements of a *business transaction use case* is represented in exactly one activity graph of a *business transaction*. A *maps to* dependency between them allow traceability between the requirements and the *business transaction*, which is defined in a *business interaction view*. In our example, the `announce waste transport` requirements (3) are mapped to a corresponding choreography (7). The same mapping is made for the `announce transport arrival` requirements (4+8).

A *business transaction* is characterized as follows: If an *authorized role* recognizes an event that changes the state of a *business entity*, it initiates a *business transaction* to synchronize with the collaborating *authorized role*. A *business transaction* is an atomic unit that leads to a synchronized state in both information systems. 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. 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 *business information* flows from the initiator to the responder to set the interim state and backwards to set the final and irreversible state change. Irreversible means that returning to an original state requires compensation by another *business transaction*.

Owing to this strict definition, 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 *business transaction 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. In our example the *business transactions* `announce waste transport` (7) is a two-way transaction. Sending the `waste movement form envelope` sets the interim state `announced` of the *business entity* `waste transport`. The reply in the `waste movement response envelope` sets it into the state `approved`. The *business transaction* `announce transport arrival` (8) is a one-way transaction which immediately sets the state `arrived` of the *business entity* `waste transport`.

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*. This one-to-one relationship is denoted by another *maps to* dependency. 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 transactio*n. 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). *Maps to* dependencies keep track of this refinement. *Business collaboration activities* - which are not used in our example - are refined by a nested *business collaboration protocol*.

In this short paper we do not further concentrate on the *business information views* which are used to define the structure of *business documents* exchanged in *business transactions*. It should be noted that each ot the three *information envelopes* exchanged in the two *business transactions* leads to a *business information view* describing the envelope's document structure (10). In the near future we will release another short paper explaining how to build business documents that are based on core components and are represented by UML class diagrams.


WasteManagement
«BusinessDomainView» Waste Management
«BusinessRequirementsView» Waste Management
«BusinessProcessView» Manage End-to-End Waste Transport
Manage End-to-End Waste Transport
«BusinessEntityView» WasteTransport
«BusinessEntity» WasteTransport
WasteTransport
«TransactionRequirementsView» Announce Waste Transport
Announce Waste Transport
«AuthorizedRole» InformationProvider
«AuthorizedRole» InformationConsumer
«BusinessTransactionUseCase» Announce Waste Transport
«TransactionRequirementsView» Announce Transport Arrival
Announce Transport Arrival
«AuthorizedRole» InformationProvider
«AuthorizedRole» InformationConsumer
«BusinessTransactionUseCase» Announce Transport Arrival
«CollaborationRequirementsView» Manage Waste Transport
Manage Waste Transport
«AuthorizedRole» OutboundRole
«AuthorizedRole» InboundRole
«BusinessCollaborationUseCase» Manage Waste Transport
«CollaborationRealizationView» Manage Waste Transport,
Manage Waste Transport
«AuthorizedRole» InboundRole
«AuthorizedRole» OutboundRole
«BusinessCollaborationRealization» Manage Waste Transport
«CollaborationRealizationView» Manage Waste Transport,
«CollaborationRealizationView» Manage Waste Transport,
«BusinessTransactionView» Waste Management
WasteManagement
«BusinessInteractionView» Announce Waste Transport
«BusinessInteraction» AnnounceWasteTransport
Announce Waste Transport
«BusinessTransactionSwimlane» Requestor: InformationProvider
«RequestingInformationEnvelope» : WasteMovementFormEnvelope
«RequestingBusinessActivity» Notify Waste Transport
«BusinessTransactionSwimlane» Responder: InformationConsumer
«RespondingInformationEnvelope» : WasteResponseFormEnvelope
«RespondingBusinessActivity» Process Waste Movement Form
«BusinessInteractionView» Announce Transport Arrival
«BusinessInteraction» AnnounceTransportArrival
Announce Transport Arrival
«BusinessTransactionSwimlane» Requestor: InformationProvider
«RequestingInformationEnvelope» : TransportArrivalFormEnvelope
«RequestingBusinessActivity» Notify Transport Arrival
«BusinessTransactionSwimlane» Responder: InformationConsumer
«RespondingBusinessActivity» Process Transport Arrival Form
«BusinessChoreographyView» Manage Waste Transport
«BusinessChoreography» Manage Waste Transport
Manage Waste Transport
«BusinessTransactionActivity» Announce Transport Arrival
«BusinessTransactionActivity» Announce Waste Transport
«BusinessInformationView» Waste Movement Form
«InformationEnvelope» WasteMovementFormEnvelope
«BusinessInformationView» Waste Response Form
WasteResponseFormEnvelope
«BusinessInformationView» Transport Arrival Form
«InformationEnvelope» TransportArrivalFormEnvelope
act Manage End-to-End Waste Transport
:Notifier
:ExportAuthority
:ImportAuthority
:Consignee
«BusinessProcessActivity» Pre-inform on waste transport
«SharedBusinessEnti... :WasteTransport [announced]
«BusinessProcessActivity» Be informed on waste transport
«BusinessProcessActivity» Respond on waste transport information
«SharedBusinessEnti... :WasteTransport [approved]
«BusinessProcessActivity» Inform of waste receipt
«SharedBusinessEnti... :WasteTransport [arrived]
uc Manage Waste Transport, Ex - ExA
Notifier
ExportAuthority
6a
«mapsTo»
«BusinessCollaborationRealization» Manage Waste Transport
«participates»
OutboundRole
InboundRole
5
«realize»
«BusinessCollaborationUseCase» Manage Waste Transport
<<maps to>>
<<includes>>
uc Announce Waste Transport
«BusinessTransactionUseCase» Announce Waste Transport
InformationProvider
InformationConsumer
3
uc Announce Transport Arrival
«BusinessTransactionUseCase» Announce Transport Arrival
4
act Manage Waste Transport
9
«BusinessTransactionActivity» Announce Waste Transport
«BusinessTransactionActivity» Announce Transport Arrival
stm WasteTransport
2
Initial
«BusinessEntityState» announced
«BusinessEntityState» approved
«BusinessEntityState» arrived
Final
act Announce Waste Transport
«BusinessTransactionSwimlane»
Requestor :InformationProvider
Responder :InformationConsumer
7
Initial State
«RequestingBusinessActivity» Notify Waste Transport
«RespondingInformationEnv... :WasteResponseFormEnvelope
Failure
Success
«RequestingInformationEnve... :WasteMovementFormEnvelope
«RespondingBusinessActivity» Process Waste Movement Form
act Announce Transport Arrival
8
«RequestingBusinessActivity» Notify Transport Arrival
«RequestingInformationEnvel... :TransportArrivalFormEnvelope
«RespondingBusinessActivity» Process Transport Arrival Form