Towards a Complete Metamodel for DEMO CM

Mark A. T. Mulder(✉)

Leusden, Netherlands
mark@mulderrr.nl

Abstract. The Design and Engineering Method for Organisations (DEMO) is the principal methodology in Enterprise Engineering (EE). The Design and Engineering Method for Organisations Specification Language (DEMOSL) states the rules, legends, and metamodel of DEMO. Therefore, any DEMO model must comply with this specification. Moreover, to enable automation of the DEMO model validation, we need a metamodel that can accurately represent DEMO models. In our research we are expanding the DEMOSL to be able to express all DEMO models and rules. This paper reports on the attempt to build this metamodel for four elements of the Construction Model (CM) using mathematical and semantic notation. The findings on the validation done on DEMOSL have been added to the build metamodel. We found the notations to be sufficient to describe and validate the DEMO (CM) models.

1 Introduction

The Design and Engineering Method for Organisations (DEMO) [1] is the principal methodology in Enterprise Engineering [2]. This so-called essential model of an organisation is the integrated whole of four aspect models: the Construction Model (CM), the Action Model (AM), the Process Model (PM) and the Fact Model (FM). Each model is expressed in one or more diagrams and one or more cross-model tables.

The CM is the first and the most comprehensive model to produce when modelling an organisation in DEMO, applying the Organisational Essence Revealing (OER) method. A CM is a model of the construction of an organisation (or rather, of a Scope of Interest), by which is understood the identified transaction kinds and the actor roles that are either executor or initiator of these transaction kinds. The resulting 'network' of transaction kinds and actor roles is always a set of tree structures, which arise from the inherent property that every transaction kind has exactly one elementary actor role as its executor (and vice versa), and that every actor role may be initiator of one or more transaction kinds, or none.

A CM is expressed in an Organisation Construction Diagram (OCD), a Transaction Product Table (TPT) and a Bank Contents Table (BCT). An OCD is a graphical representation of the identified transaction kinds and actor roles, and the links between them. Apart from initiator and executor links, actor roles

C. Debruyne et al. (Eds.): OTM 2018 Workshops, LNCS 11231, pp. 97–106, 2019.
https://doi.org/10.1007/978-3-030-11683-5_10

may also be connected to transaction kinds through information links. They express that the actor role has (reading) access to the history of all transactions of the transaction kind with which it is connected. Therefore, the shape of the transaction kind may also be interpreted as a transaction bank. This paper limits itself to four elements of a CM: Elementary Actor Role (EAR), Composite Actor Role (CAR), Transaction Kind (TK) and Aggregate Transaction Kind (ATK). In the remainder of this paper, we will discuss the DEMO metamodel by first defining the notion of Meta Model. We will subsequently report on our findings on the metamodel for the CM aspect model. We end with conclusions and suggestions for future research.

2 Research Design

This research was conducted using a focus group of professional users of the metamodel. The usage of the DEMO models as well as the requirements for the metamodel were discussed and adapted to the insights collected.

3 Extending DEMOSL

We will use the following definition of a metamodel [3]: "meta-models define sets of valid models, facilitating their transformation, serialization, and exchange." The base of the metamodel is the Design and Engineering Method for Organisations Specification Language (DEMOSL) [4]. This metamodel has been validated [3] and this validation has been taken into account in the metamodel presented in this paper. From this validation paper we know that we have to include eight issues in the metamodel: (a) the Scope of Interest (SoI); (b) interstriction ATK-CAR/SoI; (c) mandatory TK for ATK; (d) interstriction EAR to ATK; (e) CAR to TK relation; (f) TK in CAR relation; (g) CAR hierarchy; (h) mandatory EAR for CAR.

3.1 Extending the Verification Rules

The rules that control the correctness of the model within the meta model can be formulated in mathematical terms of collections. Every entity type in the meta model represents a collection, written bold in the formulas. An instance of an entity type is written in italic. Per entity type we will formulate all relations of the meta model as collections. The base formulas are listed here:

$$\begin{aligned}\textbf{construction model} = {} & \textbf{elementary transaction kind} \\ & \cup \textbf{elementary actor role} \\ & \cup \textbf{composite actor role} \\ & \cup \textbf{aggregate transaction kind}\end{aligned} \tag{1}$$

$$\begin{aligned}\textbf{actor role} = {} & \textbf{elementary actor role} \\ & \cup \textbf{composite actor role}\end{aligned} \tag{2}$$

$$\textbf{elementary actor role} \cap \textbf{composite actor role} = \emptyset \tag{3}$$

$$\textbf{aggregate transaction kind} \cap \textbf{elementary transaction kind} = \emptyset \tag{4}$$

$$\forall x : elementarytransactionkind(x) \iff x \in \textbf{elementary transaction kind} \tag{5}$$

$$\forall x : aggregatetransactionkind(x) \iff x \in \textbf{aggregate transaction kind} \tag{6}$$

$$\forall x : elementaryactorrole(x) \iff x \in \textbf{elementary actor role} \tag{7}$$

$$\forall x : compositeactorrole(x) \iff x \in \textbf{composite actor role} \tag{8}$$

$$\forall x : actorrole(x) \implies elementaryactorrole(x) \vee compositeactorrole(x) \tag{9}$$

$$\begin{aligned}\forall x : transactionkind(x) \implies & elementarytransactionkind(x) \vee \\ & aggregatetransactionkind(x)\end{aligned} \tag{10}$$

3.2 DEMO Exchange Model

The proposal of a DEMO Exchange model of [5], which is based on DEMO 2 is a good start for the exchange model. It proposes an exchange model for the FM and the CM. The XSD for CM proposes the storage of the id, name, initiator(s), executor, and information link and result type. This information is based on older DEMO specifications and, therefore, lacks important information from the current version.

We will build this XSD structure for every element type in the DEMOSL structure. The whole DEMO model has a name and every component has an internal ID. We will model this name and ID (but we will not mention this in the element explanation).

Guid

The identification of all types and kinds within the exchange model are identified with a Global Unique IDentfier (GUID). A GUID is a 128-bit number used to identify information in computer systems.

Name Conventions

The naming of the elements used in a CM is restricted by rules. One of the rules we will mention in this paper is the transaction kind name. The specification in DEMOSL [4] states that the name is built up of lower case words. This rule can be written in XSD. Note that giving no name is allowed for practical use purposes.

```
<xs:simpleType name="TransactionKindName">
  <xs:annotation>
    <xs:documentation>The transaction_kind_name are lower case words.
        </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-z]*([␣][a-z]+)*"/>
  </xs:restriction>
</xs:simpleType>
```

4 Extending DEMOSL for the CM

Different modelling techniques allow for the representation of different properties of the object that is being modelled. Only a combination of those models will come close to the representation of the whole object. Therefore, for every component of the meta models we will define

1. Mathematical rules on the collections
   (a) XSD specification for the set
   (b) XSD specification for the item
   (c) XSD specification(s) for the types
   (d) XSD specification for the diagram element
2. Data Model for the item and relations
3. OWL representation.

With this list of models, we think that the representation of the DEMO model is sufficiently complete to validate the model and exchange information to recreate the model. We remodelled the meta model according to the findings in [3].

4.1 Transaction Kind

For TK the relation with the CAR was missing in DEMOSL [4]. This relation has been added in formula 12 *containedinCAR*.

Data Model
The data model in Fig. 1 shows the TransactionSort attribute and the description. The last attribute was added by the focus group to be able to exchange meta information about the TK.

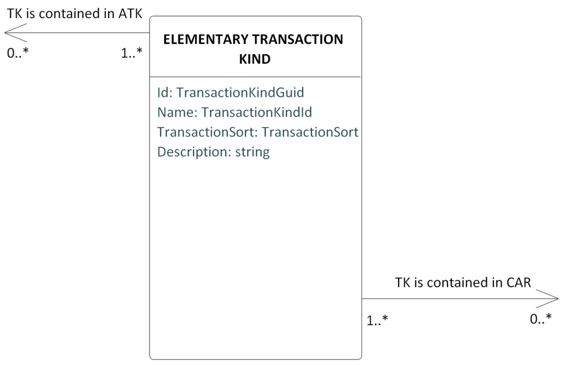

Fig. 1. TK data model

Mathematical Model

The data model relations can be shown in a mathematical way. The relations in the data model denote that a TK can be contained in both an ATK and a CAR. Based on formulas 5, 6, and 8, we can conclude 11 and 12. This last formula solves issue f [3].

$$\forall x, y : TKcontainedinATK(x, y) \implies elementarytransactionkind(x) \wedge aggregatetransactionkind(y) \tag{11}$$

$$\forall x, y : TKcontainedinCAR(x, y) \implies elementarytransactionkind(x) \wedge compositeactorrole(y) \tag{12}$$

When a TK is part of a CAR, the initiator (formula 16) and executor (formula 17) of that TK must also be part of the CAR as in formulas 13 and 14. Only then can we assure that initiation and execution of other transactions are well defined.

$$\forall x, y, p : TKcontainedinCAR(x, y) \wedge initiator(p, x) \wedge elementaryactorrole(p) \implies EARcontainedinCAR(p, y) \tag{13}$$

$$\forall x, y, p : TKcontainedinCAR(x, y) \wedge executor(x, p) \wedge elementaryactorrole(p) \implies EARcontainedinCAR(p, y) \tag{14}$$

The example in Fig. 2 shows that CA4, where T2 and T3 are part of CA4, does not comply with formulas 13 and 14 because it is missing the involvement of A1 and A3.

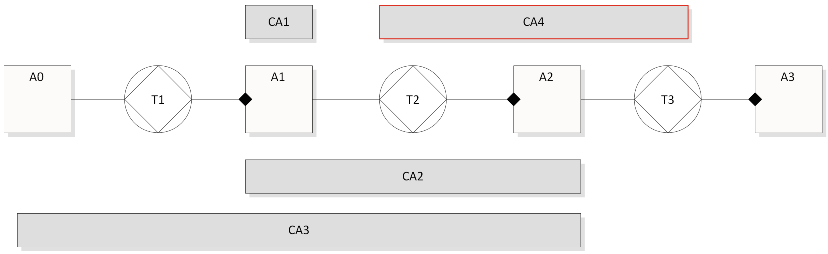

Fig. 2. TK containedinCAR example

Exchange Model

A transaction had the properties of a name, transaction sort, and an identification. We discussed in the focus group to implement the ID of the exchange data as an attribute.

```
<xs:complexType name="TransactionKind">
  <xs:sequence>
    <xs:element name="Identification" type="TransactionKindId"></
        xs:element>
    <xs:element name="Name" type="TransactionKindName"></xs:element>
    <xs:element name="TransactionSort" type="TransactionSort" default
        ="unknown"></xs:element>
  </xs:sequence>
  <xs:attribute name="Id" type="TransactionKindGuid" use="required"/>
</xs:complexType>
```

This exchange rule set described in XSD can sufficiently restrict field content for each TK.

OWL/Turtle Representation
The OWL/Turtle representation [6] allows for defining classes representing the mathematical information. We are building this representation for all entity types, attributes, relations and rules of the data model.

```
:ElementaryTransactionKind a rdfs:Class.
:EtkIdentification a owl:DatatypeProperty;
                 rdfs:subClassOf :ElementaryTransactionKind.
:EtkName a owl:DatatypeProperty;
        rdfs:subClassOf :ElementaryTransactionKind.
```

4.2 Aggregate Transaction Kind

The restrictions on the existence of ATK is contained in the TK. ATKs may exist with or without contained TKs.

Data Model
The ATK has the same attributes as the TK, except for the TransactionSort (Fig. 3).

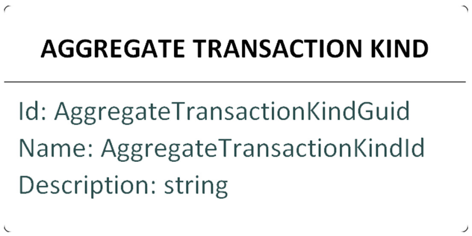
AGGREGATE TRANSACTION KIND
Id: AggregateTransactionKindGuid
Name: AggregateTransactionKindId
Description: string

Fig. 3. DEMOSL ATK metamodel

Mathematical
As can be seen in Fig. 3, no relations are coming from the ATK. Therefore, no collection formulas have been formulated for the ATK.

Exchange Model
In the exchange model the ATK is modelled explicitly. Apart from the TransactionSort element, that is left out, this exchange model is equivalent to the TK exchange model.

4.3 Elementary Actor Role

Data Model
The EAR data-metamodel has five relations to other elements (Fig. 4).

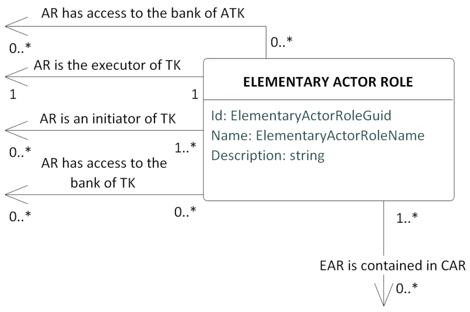

Fig. 4. DEMOSL EAR metamodel

Mathematical

$$\begin{aligned}\forall x, y :& EARcontainedinCAR(x, y) \implies \\ & elementaryactorrole(x) \wedge compositeactorrole(y)\end{aligned} \tag{15}$$

$$\forall x, y : initiator(x, y) \implies actorrole(x) \wedge elementarytransactionkind(y) \tag{16}$$

$$\forall x, y : executor(x, y) \implies elementarytransactionkind(x) \wedge actorrole(y) \tag{17}$$

$$\begin{aligned}\forall x, y, z :& executor(x, y) \wedge executor(x, z) \wedge \\ & elementaryactorrole(y) \wedge elementaryactorrole(z) \implies y = z\end{aligned} \tag{18}$$

$$\forall x, y : information(x, y) \implies actorrole(x) \wedge transactionkind(y) \tag{19}$$

Formula 18 makes sure a TK can only be executed by a single EAR.

Formula 19 solves the issues c and d from [3] and the combination of this formula with the inheritance from EAR solves issue b. The inheritance itself solves issue e and h.

The executor of an transaction kind should be an actor role. This can be an elementary actor role or a CAR. When the executor of a TK is an EAR, and this EAR is part of a CAR, the CAR is also an executor of the TK. We use the formulation of CAR and EAR relations, i.e. formula 21. As seen from the execution viewpoint of the transaction kind we get formula 20.

$$\begin{aligned}\forall x, y, z :& executor(x, y) \wedge executor(x, z) \wedge \\ & elementarytransactionkind(x) \wedge elementaryactorrole(y) \wedge \\ & compositeactorrole(z) \implies EARcontainedinCAR(z, y)\end{aligned} \tag{20}$$

Exchange Model

```
<xs:complexType name="ElementaryActorRole">
  <xs:sequence>
    <xs:element name="Identification" type="ActorRoleId"></xs:element
        >
    <xs:element name="Name" type="ActorRoleName"></xs:element>
  </xs:sequence>
  <xs:attribute name="Id" type="ElementaryActorRoleGuid" use="
      required"/>
</xs:complexType>
```

OWL Representation

```
:executor a [a owl:Restriction ;
            owl:onProperty :executor ;
            owl:onClass :ElementaryTransactionKind ;
            owl:maxQualifiedCardinality "1"^^xsd:int],
           [a owl:Restriction ;
            owl:onProperty :executor ;
            owl:onClass :ElementaryActorRole ;
            owl:maxQualifiedCardinality "1"^^xsd:int],
           owl:ObjectProperty.
```

4.4 Composite Actor Role

Data Model

In the CM, the CAR and the EAR are modelled as separate entity types. By modelling the CAR (Fig. 5) as a specialisation of EAR, the relations of the elementary actor role are also available for the composite actor role. This change allows us to model the composite actor role 'customer' in the pizza case where the customer is the initiator and executor of a TK without any elementary actor roles present.

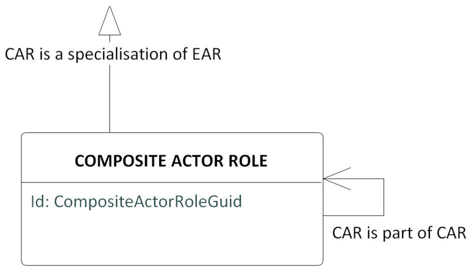

Fig. 5. DEMOSL CAR metamodel

Absent from in the DEMOSL meta model [4], is the SoI itself. We argue that the SoI is equivalent to the CAR When starting a model, the first actor is a composite actor until one is able to retrieve the information to redesign the actor roles into a white box model. The CAR does not vanish. The CAR becomes equal to the SoI as can be seen in Figs. 6 and 7. Therefore the only difference between a SoI and a CAR is its appearance in the diagram.

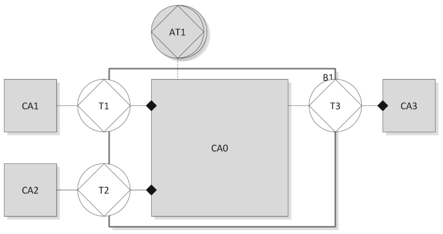

Fig. 6. CM modelling step 1

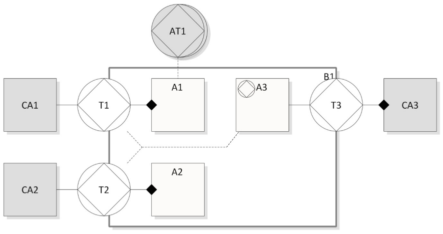

Fig. 7. CM modelling step 2

Mathematical

The *CARispartofCAR* relation describes that the CAR contains another level of CAR. This means that if a CAR is described in more detail, it is connected to the CAR it belongs to.

$$\begin{aligned} \forall x, y : CARispartofCAR(x, y) &\implies compositeactorrole(x) \wedge \\ &\quad\; compositeactorrole(y) \\ \nexists x, y : CARispartofCAR(x, y) &\implies CARispartofCAR(y, x) \end{aligned} \tag{21}$$

$$\begin{aligned} \forall x, y, z : & executor(x, y) \wedge executor(x, z) \wedge \\ & elementarytransactionkind(x) \wedge compositeactorrole(y) \wedge \\ & compositeactorrole(z) \implies EARcontainedinCAR(z, y) \end{aligned} \tag{22}$$

Exchange Model

The exchange model of the CAR is equivalent to the EAR. The type references have been renamed to match the composite intention.

5 Conclusions and Future Research

Creating a metamodel for DEMO (the CM in particular) is possible from the base DEMOSL created. The omissions found in [3] have been solved using the new data model and mathematical model. The research on OWL is still in progress. We expect it will help us finding the execution rules for automating the mathematical model.

Even though this paper does not explicitly use Fact Based Modelling (FBM) notation, the next topic in DEMO that is going to be meta modelled is the FM. Originally the notation of Object Role Modeling (ORM) has been used in DEMO 2 to express the facts in a graphical way, including the instantiation of the model with examples. The metamodel of the CM is, in fact, modelled using ORM, and simplified using the DEMO 3 notation.

In practice, modelling the CM is done using the same method as the one used for modelling the data. Therefore, part of the research is evaluating the amount of benefit by using FBM methods.

We will further expand this metamodel to cover the whole of DEMO and to be able to exchange all information currently used in DEMO models around the world.

References

1. Dietz, J.L.G.: Enterprise Ontology: Theory and Methodology. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-33149-2
2. Dietz, J.L.G., Hoogervorst, J.A.P.: The discipline of enterprise engineering. Int. J. Organ. Des. Eng. **3**, 86–114 (2013)
3. Mulder, M.A.T.: Cross channel communication design critical literature review. In: Aveiro, D., Pergl, R., Gouveia, D. (eds.) EEWC 2016. LNBIP, vol. 252, pp. 166–180. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39567-8_11
4. Dietz, J.L.G., Mulder, M.A.T.: Demo specification language 3.7 (2017)
5. Wang, Y., Albani, A., Barjis, J.: Transformation of DEMO metamodel into XML schema. In: Albani, A., Dietz, J.L.G., Verelst, J. (eds.) EEWC 2011. LNBIP, vol. 79, pp. 46–60. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21058-7_4
6. Hitzler, P., Krotsch, M., Parsia, B., Rudolph, S.: OWL 2 Web Ontology Language Primer, 2nd edn. (2012). https://www.w3.org/tr/2012/rec-owl2-primer-20121211/