Evolving the DEMO Specification Language

Mark A. T. Mulder[1,2] [0000−0002−1846−0238] and
Hend*erik* A. Proper[3,4] [0000−0002−7318−2496]

[1] TEEC2, Hoevelaken, the Netherlands
[2] Radboud University, Nijmegen, the Netherlands
[3] Luxembourg Institute of Science and Technology (LIST), Belval, Luxembourg
[4] University of Luxembourg, Luxembourg
markmulder@teec2.nl, e.proper@acm.org

Abstract. This paper reports on the current state of the DEMO Specification Language (DEMOSL). The Design and Engineering Methodology for Organisations (DEMO) is a principal methodology in Enterprise Engineering (EE), while the DEMOSL defines the accompanying integrated modelling landscape.
DEMO provides a method to produce so-called essential models of organisations, which are highly abstracted ontological models. For the DEMOSL this implies that it should enable the integration of the different models used in organisations when they apply DEMO, while also enabling tool support, visualisation, as well as the exchange of models. This paper describes the state of the meta-models, as referenced in the MU-theory, for the modelling and visualisation of DEMO models. The purpose and examples of six of these models and meta-models are discussed, along with extensions of the visualisation of the DEMO models. Moreover, modelling rules are presented that guide the modeller and tool developer in creating diagrams and tables with their respective elements and connections.

1 Introduction

The aim of this paper is to report on the current state of the DEMOSL. The DEMO [3] is a principal methodology in EE [4], while the DEMOSL defines the accompanying integrated modelling landscape.

DEMO provides a method to produce so-called essential models of organisations, which are highly abstracted ontological models. The first step in applying DEMO is producing the so-called essential model of an organisation. An essential model comprises 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 of these models is expressed in terms of one or more diagrams and accompanied by one or more cross-model tables.

The DEMOSL defines the accompanying integrated modelling landscape. As such, it should allow for the integration of the different DEMO models used in organisations when applying DEMO, while also enabling tool support, visualisation, as well as the exchange of models.

The DEMOSL, as discussed in this paper, involves a further elaboration and refinement of the earlier version reported in [9]. The elaboration and refinements are, amongst others, based on experiences with the establishment of an integrated tool environment which supports the creation and management of DEMO related models [10].

The remainder of this paper is structured as follows. Section 2 will briefly visit the research methodological background of the research project. Since DEMO's modelling landscape is largely founded on the underlying Model Universe (MU) theory [3], Section 3 will summarise the relevant elements of this theory. Using this as a background, Section 4 provides an overview of the different models that play a role in the DEMO modelling landscape. Section 5 is dedicated to the discussion of the meta-models and how they have been used in the current tool environment. This is followed (in Section 6) by a reflection n the experiences with the implementation of the meta-models in an operational tool environment. This discussion will also highlight some of the refinements and improvements that had to be made in DEMO's meta-model and earlier versions of the DEMOSL to make them operationalisable in a tool environment. Finally, before concluding, Section 7 provides an outlook for future research.

2 Research background

The research effort as reported on in this paper, aims to improve the DEMOSL specification. The DEMOSL is available in a number of versions and at the moment the reported research project commenced, the latest version was 3.7. Moverover, this version had already undergone an initial validation [9].

Originally, the DEMOSL was designed to better understand the concepts that need to be included in DEMO models. However, for the development of automated tool support for DEMO, more details and specificity are needed than is included in the original DEMOSL. This includes e.g. details about the graphical layout of diagrams, exchange of models between tool environments, as well as the specificity of the actual models. As such, the goal is the creation of a version of the DEMOSL that is complete enough to automatically validate the rules and restrictions involved in DEMO modelling using automated tools.

The resulting specification is divided into four kinds of models, and four associated meta-models. These four model kinds need to comply with the requirements as stated in [10]. The required meta-models for the tooling have been iterated to specifically fulfil the following three requirements (see [10]):

1. The tool must support the interchange of models.
2. The tool must support the creation of all four aspect models of DEMO.
3. The tool must allow for model verification against the DEMO meta-model, DEMOSL.

Next to these requirements, the meta-models need to be "build" on top of the MU theory, which is a foundational part of DEMO [5] and was known in a previous iteration of DEMO as *Factual Knowledge* [3].

From a research methodological perspective, the research effort as reported on in this paper, uses the Design Science Research (DSR) approach [1, 12]. In design science terminology, the DEMOSL is an artefact. In applying the design science research approach, the development of the DEMOSL is rooted in the existing EE and DEMO body of knowledge (enabling *rigour*), while allowing for the iterative improvement and refinement of the artefact in terms of experiments and cases (enabling *relevance*). The latter involve(d) experiments in terms of the implementation of DEMOSL in an enterprise-grade tool environment [10], as well as the application in real-world cases.

3 MU theory

The MU-theory provides the theoretical foundation of the notions of model, modelling, and modelling language as used in DEMO. The most recent version of the MU-theory has been presented in [5].

Given the aim of the reported research to further evolve the accompanying DEMOSL, the MU-theory is considered as a fixed and pre-defined part of the knowledge base (in design science terms). In this section, we highlight some of the key elements of the MU-theory that are relevant to the understanding and positioning of the remainder of the paper.

The MU-theory theory adopts Apostel's [2] definition of model: "*Any subject using a system A to obtain knowledge of a system B, is using A as a model of B.*" This definition conveys the basic understanding of the notion of model as being a role [5].

A key part of the MU-theory is the General Conceptual Modelling Framework (GCMF), as depicted in Fig. 1. The MU-theory uses the term complex to refer to systems in the general sense, as used in Apostel's [2] definition of model. However, since DEMO uses its own (more specific) definition of system, the term *complex* is preferred when speaking about modelling in general (as is the case for the MU-theory).

The basic thinking underlying the GCMF is based on the semiotic triangle by Ogden and Richards [11]. In line with this, the MU-theory refers to the phenomena we observe and interact with in reality as *concrete complexes*, while the conception of these complexes an actor harbours in their mind is referred to as the *conceptual complexes*, and a resulting symbolic representation (such as a diagram, a narrative description, or an XML document) as *symbolic complexes*.

When combining this with Apostel's definition of a model, then one can state that when such a complex A is used to obtain knowledge of a complex B, then A is said to be a model of B. As a corollary to this, one can (using the terminology from the MU-theory) distinguish between *concrete models*, *conceptual models* and *symbolic models*.

The MU-theory states that the creation of a conceptual complex needs a prescription, called a *conceptual schema*. It is furthermore assumed that a *conceptual complex* can only be created in the mind of some actor observing a domain, if they already have a *conceptual schema* in terms of which they can

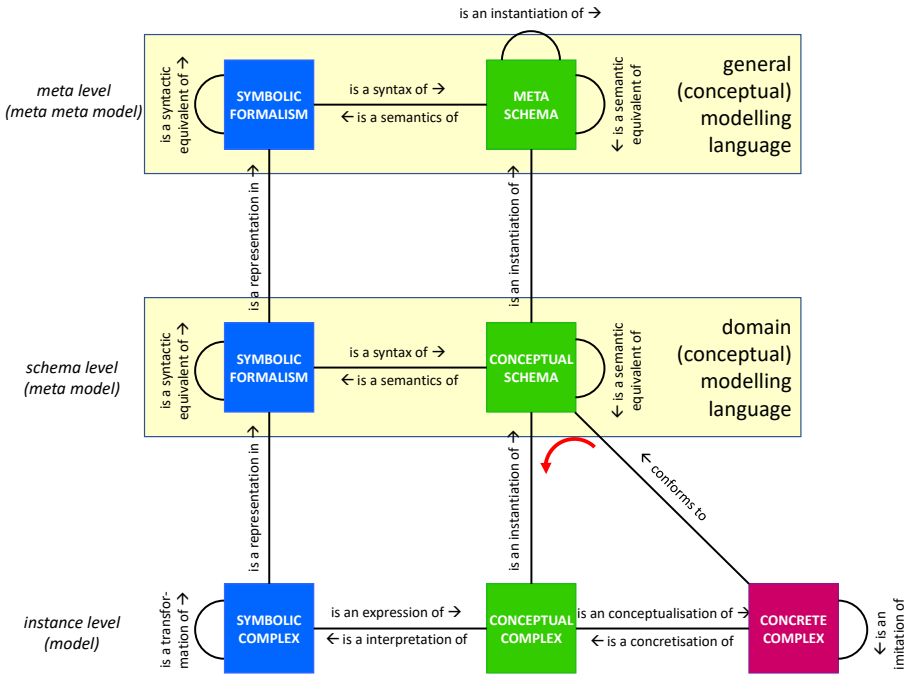

Fig. 1: The General Conceptual Modelling Framework, adopted from [5]

observe the world. This is indicated by the red curved arrow in Fig. 1. As such, a *conceptual schema* limits the observable world (of an actor). The MU-theory also states that a modelling language essentially involves a combination of a conceptual schema and a corresponding symbolic formalism that provides the symbolic representation of the conceptual schema.

The DEMOSL is targeted at the schema level of Fig. 1. In other words, it should provide a symbolic formalism corresponding to the conceptual schema(s) used by the different model (and diagram) kinds within DEMO. As illustrated in Fig. 1, the MU-theory also identifies a meta level involving a meta schema and a corresponding symbolic formalism. This is where, in our case, we will find UML class diagrams and XML Document Type Definitions (DTDs) to define the DEMOSL.

The MU-theory also provides global guidelines on how to create the various levels of complexes and schemas. Finding a lower level from a higher level can be accomplished by deduction, which also provides a good method for the verification of models. Induction can be used to derive the meta level from examples. In doing the latter, it can be helpful to transform graphical languages into structured textual languages, called verbalisation. We used both deduction and induction methods to create meta-models and DEMO information models which we will describe in the next sections.

4 Perspectives on DEMO models

Before we proceed to a discussion of the meta-models involved in the DEMOSL, we need to distinguish between three perspectives on the models used when applying DEMO. These are illustrated in Fig. 2.

The first perspective is the *methodology perspective*. Even though it provides a thorough theoretical basis, the DEMO methodology book [5] was primarily written to teach learners (students and practitioners) to create models in accordance with the DEMO way of thinking, and draw (human to human) communicable models to reason about the organisation. Since the book puts the priority on "doing", when introducing the different model kinds, there was no need for a strict meta-model. Furthermore, the formalisation(s) provided in the book aim to support didactic goals rather than the development of automated modelling tools. As such, it was never meant to provide a detailed formalisation and meta-models, that would enable the development of, and automated support for, the methodology. As discussed in [7], different meta-modelling and formalisation goals will / should also result in formalisations with different level(s) of detail / specificity.

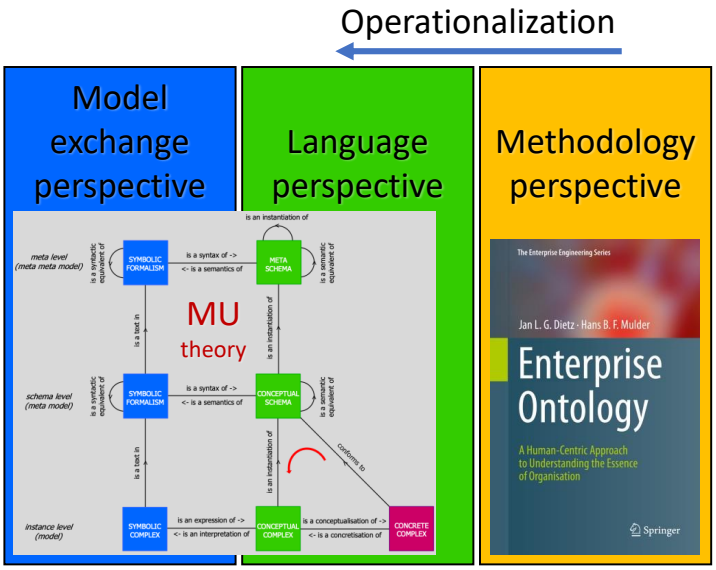

Fig. 2: DEMO Model Perspectives

When using DEMOSL as base for tool development, then it should indeed provide a more complete and detailed formalisation and meta-model; taking us to the *language perspective*. This is the perspective where it should be possible to (automatically) reason about models and meta-models, to e.g. support model verification. This requires the formalisation of the language definition in a format that lends itself better for automatic processing [7]. It also requires the completion and operationalisation of the ontological meta-models as included in the

methodology perspective; in particular the rules and constraints to be applied to the actual models.

The last *model exchange perspective* concerns the exchange and storage of model information. Reasoning about models is only possible when everybody has the same notation of the model and the notation enables the reasoning in the meta-model.

The notation used must be the same across different modelling tools. Therefore, the meaning of the elements and attributes must be well defined. Moreover, the notation must enable the reasoning within the exchange model to be able to check consistency of this model. In addition, the internal consistency needs to be correct and needs to be checked before the model can be converted back to the language perspective.

5 Overview of the meta-models

In this section the new meta-models will be presented. These meta-models add information, structure and completeness to the existing version of DEMOSL. Our research concluded that without these improvements the automatic validation and exchange of DEMO models using tooling is not possible.

Figure 3 shows the landscape of meta-models that together have the capability of modelling organisations using DEMO. The landscape is depicted as a three dimensional framework, projected on top of the framework from the MU-theory.

The two levels that have been projected on top of the framework from the MU-theory correspond to the distinction as discussed in Section 4.

The language perspective is the part that is implemented in terms of software, in order to reason about the (ontological) model itself, the way it may be exchanged, as well as how these could /should be visualised.

The visualisation perspective is concerned with the way an actual DEMO model is to be visualised in terms of diagrams. This leads to a visualisation model conform a visualisation meta-model. A visualisation model involves a set of visualisation attributes (position, size, colours, etc) on top of an (visualised part of) ontological model. The visualisation meta-model describes these attributes and contains the scripting and data structure of the diagrams. Restrictions on the allowed elements, attributes, and connections in a diagram are also formalised in the visualisation meta-model.

Below, we discuss all mentioned models and meta-models from right to left, from the top to the bottom of Figure 3

5.1 High Level Ontological meta-model

The high level ontological meta-model of DEMO is depicted in Fig. 4. The diagram shows the existing property types and concepts in black. The new property types and concepts that result from this research have been added in red. This

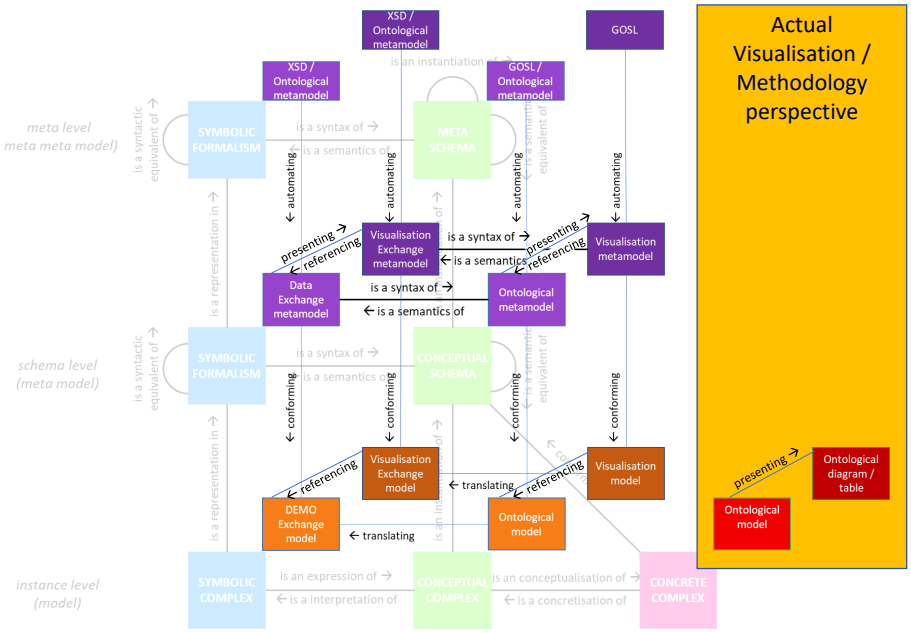

Fig. 3: Meta-model set

high level ontological meta-model is the base for the ontological meta-model of Section 5.2.

The high level ontological meta-model is the meta-model that is closest to the meta-model that lists all ontological principles of DEMO models [3]. For example, ontologically, the Composite Actor Role (CAR) cannot be an initiator of a transaction kind because an underlying elementary actor role must be the initiator. Therefore, this property type does not exist in the high level ontological meta-model.

5.2 Ontological meta-model

In contrast with the high level ontological meta-model, the ontological meta-model (see Fig. 5) contains all implementation attribute types and property types for the DEMO meta-model. The ontological meta-model also includes property types between the concepts.

The example mentioned in Section 5.1 about the CAR not being an initiator is not valid in this ontological meta-model. Whenever one designs a CAR that initiates the transaction, this property type instantiation must be present in the model and needs a representation in the ontological meta-model. Therefore, the ontological meta-model also has the property type 'AR is an initiator of TK' from the CAR to the Transaction Kind (TK). Furthermore, mathematical rules have been designed to make sure only the correct property type instantiations can be present in the final ontological model.

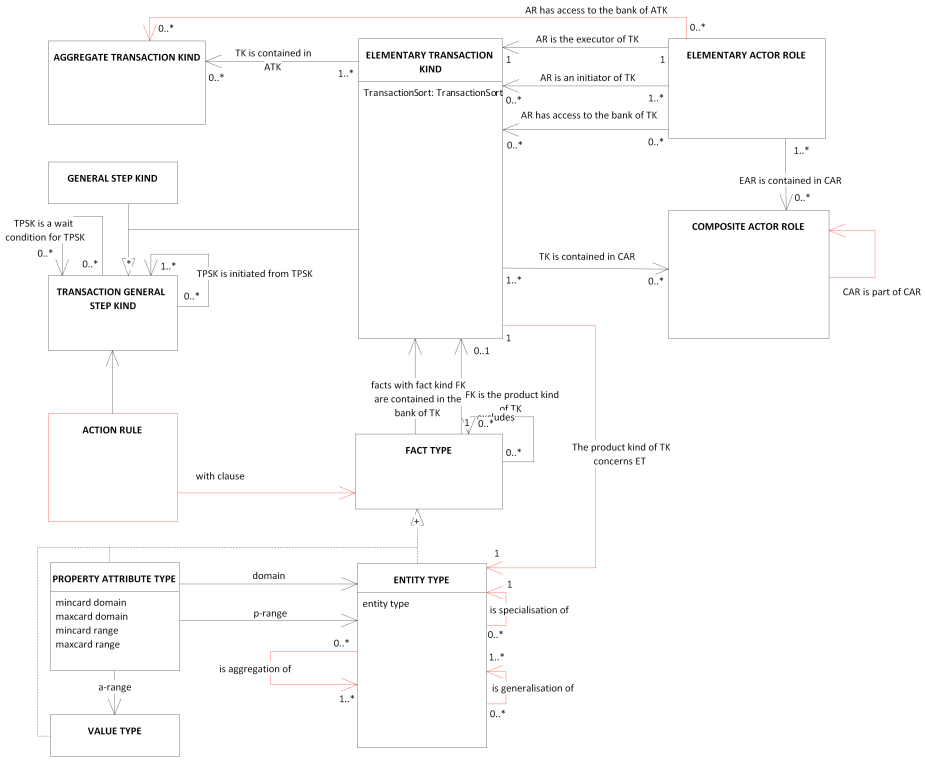

Fig. 4: High Level Ontological meta-model

The table shown in Table 1 provides all allowed Property Types between the objects of this meta-model.

5.3 Data Exchange meta-model

The concept of a data exchange meta-model is intertwined in the concept of the ontological meta-model. We describe the data exchange meta-model in terms that a computer can understand.

We choose to represent the data exchange meta-model in XML Schema Definition (XSD), which is a commonly used technique. The alternative structure, XML Metadata Interchange (XMI), has no broad implementation and, therefore, we rejected it as a candidate. The data exchange meta-model represents all entity types and property types of the ontological meta-model. This data exchange meta-model is unambiguously readable for a computer. The proposal of a DEMO exchange format [13, 14], based on DEMO 2, is a good start for our exchange meta-model. Despite the current version of DEMO, at moment of writing, is version 4, the modelling itself has not changed and is equivalent to DEMO 2. No attempt has been done to upgrade the proposed formats to DEMO 3 before. Above all, we have based our research at DEMOSL version 3.7, and, therefore, this version is an upgrade of the existing format. DEMO 4 introduces

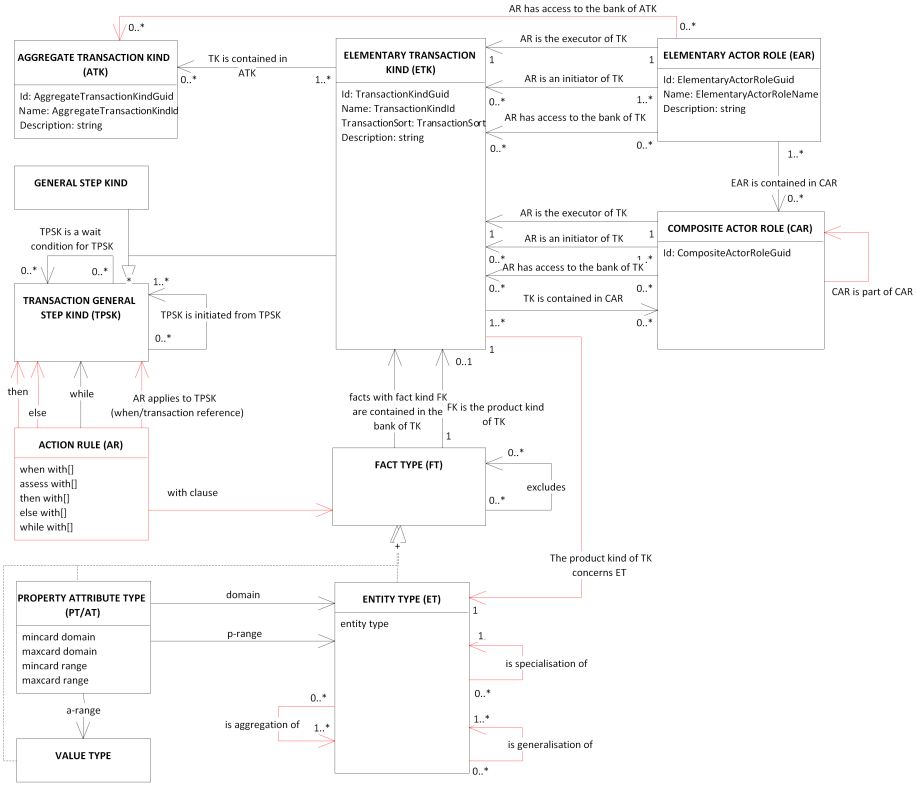

Fig. 5: Ontological meta-model

a new element but does not invalidate DEMO 2 or DEMO 3 models and, thus, the choice for these older models are still valid. In future research this exchange model has to be adapted to the features of DEMO 4. The existing paper [14] proposes a data exchange meta-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 essential information from the current version. We will build this XSD structure for every element type in the new DEMOSL proposal structure.

The first structure is the name element. The whole DEMO model has a name. Every component has an internal id. We will not mention these attributes while using the data meta-model, but we will model it.

The next structure is a Global Unique IDentfier (GUID) element. A GUID identifies all types and kinds within the exchange model and is a 128-bit number used to uniquely identify information in computer systems.

Another structure are rule elements. Rules restrict the naming of the elements used in e.g. a CM. One of the rules we will mention in this paper is the transaction kind name. The name is built up of lower case words as specified in DEMOSL

To → From ↓	ETK	ATK	EAR	CAR	ET	CET	TPSK	AR
ETK	–	c	e	ce	–	–	–	–
ATK	–	c	–	–	–	–	–	–
EAR	ia	a	–	c	–	–	–	–
CAR	ia	a	–	c	–	–	–	–
ET	o	–	–	–	xsrg	–	–	–
CET	o	–	–	–	–	–	–	–
TPSK	c	–	–	–	–	–	iy	tlwh
AR	–	–	–	–	W	–	–	–

Property Types
[c] contained in
[o] concerns
[i] initiator
[e] executor
[a] access to bank
[s] specialisation
[r] aggregation
[g] generalisation
[t] then
[l] else
[w] while
[h] when
[W] with
[x] excludes
[y] wait
[f] role of

Table 1: Element Property Types

[6]. The XSD notation enables enforcing this rule. Note that we allow giving no name for practical use purposes.

Listing 1.1: XSD example: Transaction kind

```
<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. The full set of XSD specification is already available for evaluation purposes to some research groups. The full specification with examples will be available to the public at the end of the research program. More information on the location of this specification will be published on multiple sites [5].

The CAR is the embodiment of multiple actor roles at once. This concept allows for modelling the unknown actor role or the 'don't want to know' actor role. In the DEMO methodology the concept of Scope of Interest (SoI) is used to restrain the modelling effort to a selected portion of the organisation. Absent from the DEMOSL meta-model [6], is the SoI itself. To be able to model this concept it could be added, but closer examination of the concept reveals that CAR represents the same information and is, therefore, used for this concept.

[5] http://demo.nl and http://teec2.nl

5.4 Visualisation meta-model

Communicating a graphical representation of a DEMO aspect model is not trivial. We could choose to communicate the image format in BitMaP image file (BMP), Portable Network Graphics (PNG) or Scalable Vector Graphics (SVG) formats. These formats would give the model interpreter a visualised representation, but would lack the underlying linked model information. We could also transform the graphical representation into a standard commercial format like Microsoft PowerPoint XML Presentation (PPTX) or Microsoft Visio XML Drawing (VSDX), but that could cause vendor lock-in. Instead of these formats, we chose to create only the essential attributes of the kinds and types of the diagram. This point of view is new compared to DEMOSL [6], which only describes visual properties and the exchange format [14] that does not address any visuals.

When we represent elements in a visualised graphical format, the location of the element on the diagram is an essential property. The location type attribute is available for purposes that might differ between software implementation. Next, the size, on the other hand, is used to visualise elements within a diagram. Finally, we define a line as a visualised connection between two or more points. It has a starting point, an endpoint and optionally several midpoints to create a path.

The table shown in Table 2 provides all allowed Property Types between the objects of this meta-model, while below we discuss more specific visualisation details of the various aspect models.

Construction Model – The Organisation Construction Diagram (OCD) is a diagram that expresses a part of the CM. More specifically, it contains the TK, Aggregate Transaction Kind (ATK), Elementary Actor Role (EAR) and CAR concepts. The visualised property types are the initiator, executor and access property types (see Table 2).

The representation of the OCD is used for the communication of the CM. The OCD allows TK, ATK, EAR, CAR as valid elements and initiator, executor and interstriction as valid property types.

The visualisation of the transactions and their products is a table with four columns called the Transaction Product Table (TPT). The transaction kind identification, the transaction kind name, the product kind identification and the product kind formulation. The first two columns can be filled by transaction kind attributes Identification and Name. The last two columns are the properties Identification and Name of the Independent Fact Kind. These two elements have a one-on-one relation and always exist.

We argue that *the concept of SoI is equivalent to the concept of CAR within modelling the CM.* When starting to model an organisation, the first actor inside the SoI is a composite actor role. The methodology describes that this actor role stays in place until one is able to retrieve the information to redesign the internal actor roles of this composite actor role into a white-box model. The CAR does not vanish. The CAR becomes equal to the SoI [9]. Therefore, the only difference

Diagram → Entity Types ↓	OCD	PSD	TPD	OFD	ARD	RHD	AFD	..
ETK	X	X	X	X	–	–	–	–
ATK	X	–	–	–	–	–	–	–
EAR	X	X	–	–	–	X	X	–
CAR	X	X	–	–	–	X	–	–
ET	–	–	–	X	–	–	–	–
CET	–	–	–	X	–	–	–	–
TPSK	–	X	X	–	X	–	–	–
AR	–	–	–	–	X	–	–	–
Property Types ↓								
[c] contained in	–	–	–	–	–	X	–	–
[o] concerns	–	–	–	X	–	–	–	–
[i] initiator	X	X	–	–	–	–	–	–
[e] executor	X	–	–	–	–	–	–	–
[a] access to bank	X	–	–	–	–	–	–	–
[s] specialisation	–	–	–	X	–	–	–	–
[r] aggregation	–	–	–	X	–	–	–	–
[g] generalisation	–	–	–	X	–	–	–	–
[t] then	–	–	–	–	X	–	–	–
[l] else	–	–	–	–	X	–	–	–
[w] while	–	–	–	–	X	–	–	–
[h] when	–	–	–	–	X	–	–	–
[W] with	–	–	–	–	X	–	–	–
[x] excludes	–	–	–	X	–	–	–	–
[y] wait	–	X	–	–	–	–	–	–
[f] role of	–	–	–	–	–	–	X	–

Table 2: Diagram Entity and Property Types

between an SoI and a CAR is its appearance in the diagram. The boundary transactions of the SoI are the interacting transactions in the diagram.

One could still argue that an SoI could be smaller than all transactions that interact with the CAR. This can be illustrated with an extra transaction to CA0 in the construction model. The question that remains is whether the CAR, used as SoI is the same in the diagram as in the model. In practice the SoI never exceeds the diagram boundaries and therefore, de-facto, the CAR can be used for the same purpose. When using the CAR as SoI some nice features for the hierarchy of actor roles appear.

This being said, combined with the ability to nest CARs it becomes clear that an EAR within multiple CAR is not modelled as a construction. It is possible to model this hierarchy as a functional structure, though this is not ontological. *Process Model* – In the Process Structure Diagram (PSD) the initiation of the first request is done using the relation 'is initiated from'. This would be sufficient if a step is always initiated from the same step. When, instead of the promise-step, another step is used for initiation this cannot be modelled. Therefore, the

relation has to be remodelled as a self-reference in Transaction Process Step Kind (TPSK) 'TPSK is initiated from TPSK'.

The PSD is a diagram that contains elements from the PM. More specifically it contains the TK, EAR and CAR. The visualised relations are the call-link and the wait-link relations. The Transaction Pattern Diagram (TPD) is a diagram that contains elements from the PM. More specifically it contains the TK and TPSK. The visualised relations are the process order, call link and the wait link relations (see Table 2).

Fact Model – The DEMOSL ontological model has some concepts that have made it directly to the ontological model of the fact model. Fact Type and Entity Type have been added to the data model in the same form. Other concepts will be removed or reduced because of the following reasoning. First, the Constructed Entity Type is a concept that allows reasoning about the specialised entities derived from an Entity Type. In a less conceptual model this results in the Entity Type like the generalisation relation between Constructed Entity Type and Entity Type already suggests. The same holds for a generalisation and aggregation of this constructed entity type. Both relations end in the Entity Type Set which, in turn, is an Entity Type concept on its own. This leaves the three relations as self-relations on the Entity Type. Next, the Event Type concept is a data perception of the process events of transaction kinds and TPSK. Events are related to C-acts and C-facts and only concern a concept in the P-world. Therefore, the concerns relation is created between the elementary transaction kind and the Fact Type concept.

The Object Fact Diagram (OFD) is a diagram that contains elements from the FM. More specifically it contains the Independent P-Fact Kind (IFK), Entity Type (ET) and Attribute Type (AT). The visualised relations are the reference, specialisation, aggregation and the concerns relations (see Table 2).

Action Model – The AM is the aspect model with the largest number of details about the enterprise. These details build on the CM, PM and the FM. Therefore, references to these aspect models should be used in order to keep the AM consistent with the other aspect models. Traditionally the Action Rules Specification (ARS) is specified in a natural language, restricted by a partially grammar. This allows for freedom of expression while being readable by humans. Though the advantage of the readability is large, the disadvantage of the not-strict natural language makes it difficult, and at this moment even impossible, to do an automated model verification and validation. Where the three other aspect models always have been described in a graphical way, the AM has been described within a grammar construction. This complicates communication towards different users of these models. All aspect models should have a textual and graphical representation of at least the most essential features of the model.

We added a graphical representation to the AM and improved the grammar of the ARS. All grammar references to other elements of the DEMO model have been connected in the action model.

The AM is the least developed and used aspect model of DEMO. For automation we need this model to represent all details about business rules and their relations to facts and processes.

The AM has no graphical representation in DEMO 3.7. One might argue that the separation of the event part, assess part and result part are actual AM graphical representations due to the fact that these sections of the grammar are often visualised with a background colour in educational slides. Nevertheless, these representation is not more or less that the colouring of a grammar in a modern integrated developing environment. The existing AM has four essential features.

1. Linked to a TK, it needs parameters (with-clause) to verify that all essential information is present to perform the step.
2. Linked to another step, it has to wait for other TKs to finish.
3. Inner conditions have to be checked.
4. Based on the conditions the linked follow-up actions have to be triggered.

Except for the feature three, the features have a connection to other parts of the model. Connections can be visualised as a line, like it has been done in the other three aspect models.

Let us define the connections for the AM.

- With – The with-clause connects several attributes of entities (from the FM) to either the agendum-clause, while clause of action clause. These connections will be represented with an solid line with a winged arrow head. The with connection can connect to the when, while and action clauses.
- While – The while clause connects a different transaction step (from the PM). This connection will be represented with a dotted line with a solid arrow, just like the wait link in the PSD.
- Action – The action clause connects a transaction step with another step. This is the same connection as a call link in the PSD.

Figure 6 gives an impression of the visualisation of this Action Rules Diagram (ARD). While not all arrows have been drawn to the right specifications, this early representation does give a instantiated view on the meta-model.

When we follow the example of Fig. 6, and start at the TPSK1 placed on the top of the diagram, we can see the process unfolding. For the TPSK1 the Action Rules Kind (ARK) ARK1 is the action rule that needs to be evaluated. In the when clause the required information can be expressed as entity and attribute types. In the following step, the while clause, the process can wait on the completion of the TPSK 5. When these steps have been completed the assess clause is evaluated whether the conditions, that reference the information in ET2, are true. When these conditions are evaluated the then clause triggers TPSK2 and the ARKs 2 and 6 are evaluated.

Action Rule Specification – The information in the existing AM is not detailed enough to validate a model. After analysing the existing, published action specifications we created a grammar that closely represents the existing examples and matches the wishes of the expert group that helped to create the exchange model. The verbalisation used in the DEMOSL grammar of the AM can also be

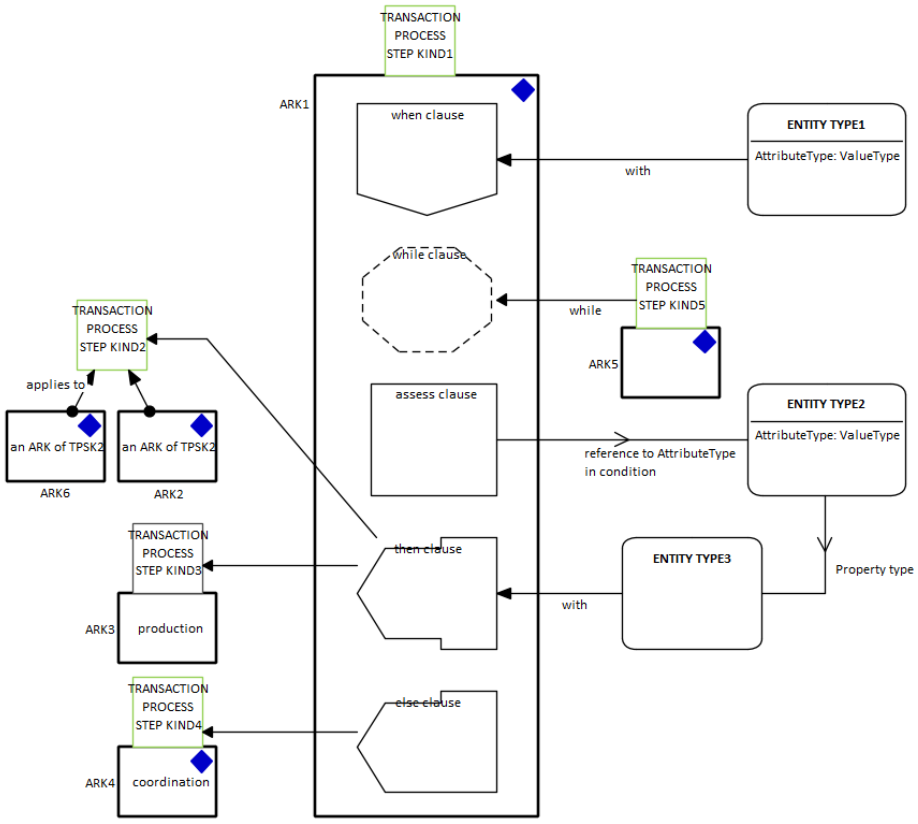

Fig. 6: ARD representation example

specified in relations to the other aspect models. These relations are described in the ontological model.

5.5 Visualisation Exchange meta-model

All element that have been described in the previous section have a visualisation element in the exchange meta-model. The structure of the visualisation elements is similar to the data exchange meta-model structure. All visualisation elements reference the data elements in the same model. The visualisation elements add the visualisation attributes for the specific diagram to that information (e.g. size, location, waypoints)

5.6 Models and exchange models

Due to the origin of the research topics, publishing created models is not possible. One of the cases was modelling dutch NEN norms that confirmed that this type of organisation can be modelled using DEMO (Fig. 8).

```
<xs:complexType name="TransactionKindElement">
  <xs:sequence>
    <xs:element name="ReferenceId" type="TransactionKindGuid"/>
    <xs:element name="Initiator" type="xs:boolean"/>
    <xs:element name="Location" type="Location"/>
    <xs:element name="Size" type="Size"/>
  </xs:sequence>
  <xs:attribute name="Id" type="DiagramElementGuid" use="required"/>
</xs:complexType>
```

Fig. 7: Visualisation Exchange TransactionKindElement

Exchanging DEMO models with other tools also needs other tools with the same meta-model implementation. The only tool that has this meta-model in place is a gamification tool. This information exchange has been tested successfully (Fig. 9).

6 Reflection

The research effort as reported on in this paper, took the description of DEMOSL 3.7 as its starting point. In DSR terminology, the meta-model that was created from this specification was the first iteration of the DEMOSL artefact. Using all model from the DEMO book [3], and associated course material, we created a second iteration.

Based on the meta-model of the second iteration artefact, a tool was created [10], which has been used in practice to model organisations. In doing so, all missing modelling elements have been added to the model and to the tool and used in successive cases. After more than seven real live cases we are convinced that the CM, PM and FM are fairly complete to hold all elements and property types needed for modelling DEMO.

The AM was not complete enough, but has improved a lot compared to the original version. These are not only elements that could be missing in the tool and the meta-model, but often challenge the theory as well because the obvious parts of DEMO can be modelled with more ease than before.

A major challenge in DEMO visualisation remains the potential variety of stakeholders [8]. The more types of stakeholders the larger number of viewpoints might be needed. To aid in bridging the gap between stakeholders in practice, we added a functional concept to the construction model. This is the functional value that the organisation gives to the construction model in its organisation. This one-on-one functional translation is the first step to connect the functional business and construction domains of DEMO.

We also did our first steps in developing the eXtended Organisational Essence and Revealing (XOER) method. Modelling all viewpoints at the same time looks promising as it helps modelling and thinking.

During practical use of DEMO models, we encountered some diagrams that were needed to fully represent the model. First, there is the Actor role Function Diagram (AFD). This diagram can be represented in an Actor role Function

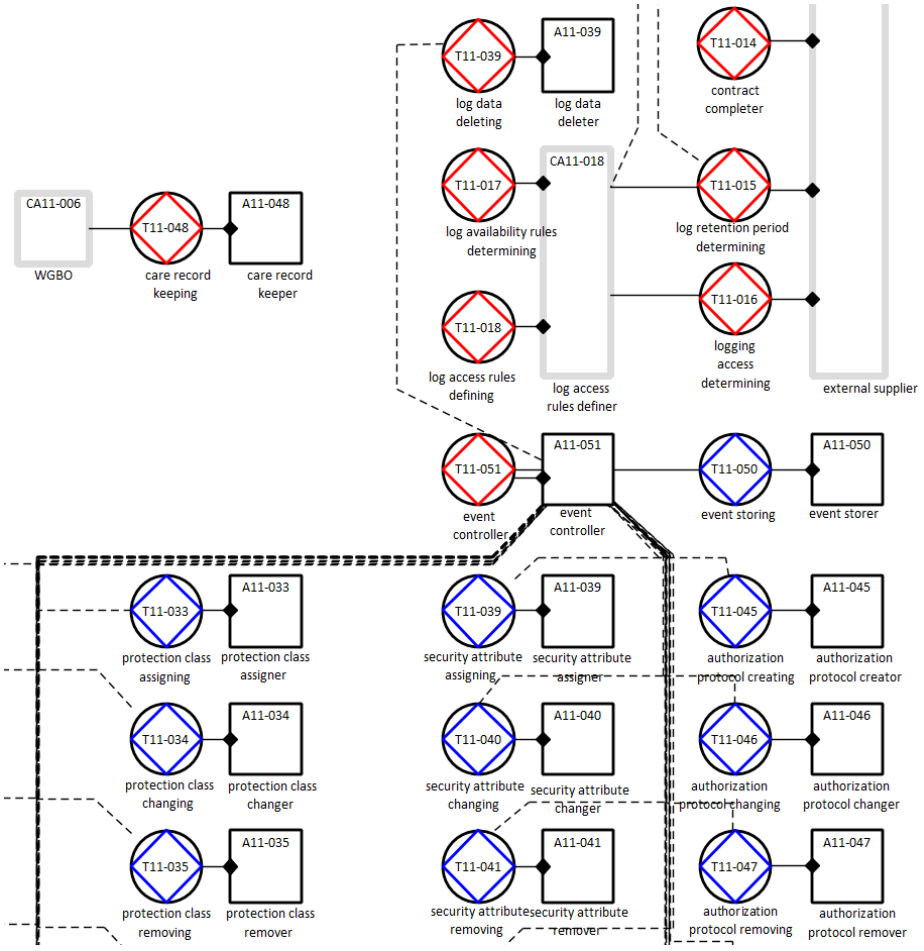

Fig. 8: Partial NEN example

Table (AFT), but that representation has limited readability. The AFD is the graphical equivalent of the AFT.

The AFD is a diagram that contains elements from both the CM and the implementation world. More specifically it contains the EAR and CAR.

Summarizing these reflections, we made adjustments in the DEMOSL to reflect practical needs in DEMO modelling and to enable automation of the modelling and validation of these models. Furthermore, we added steps in the methodology to cover a wider perspective of the modelling needs that do not necessary belong to the ontology domain, but surely are adjacent and often more visible to business stakeholders. We find the extensions to DEMOSL to be relevant and useful.

Fig. 9: Gamification Scene

7 Future Research

The upcoming iteration of XOER, on which the work has been started as we write this paper, will help us to make this method describable and educable. It will also be subject to an expert group for validation of the meta-model and the used representations.

In a parallel track we are also adding the concepts that are needed for the implementation of the (DEMO-based) VISI[6] standard for the Dutch construction sector; which are now part of the current meta-model.

More research is needed on the visualisation of the created DEMO models. A current project is used to investigate the influence of projecting well-known concepts on the explanation and visualisation of DEMO concepts. Although these concepts might not be completely correct, it might improve the acceptability of DEMO as a methodology.

8 Conclusion

In this paper, we discussed the current state of the DEMOSL. In the context of a design science based research effort, the DEMOSL is being refined and extended such that it can serve as a base for automated tool development. We have now reached a stage where the meta-model is sufficient to automate the modelling and validation of the model. Furthermore, the extension of the DEMOSL makes it more comprehensible for people to create information models that support modelling in or with DEMO.

As a next step the meta-models will be extended even further to completely support DEMO 3 and the successor DEMO 4. Development on the methodology needs support in the meta-models, information and automation.

[6] `https://www.crow.nl/downloads/pdf/contracteren/visi/visi-systematiek-doelstellingen-grondbeginselen-be.aspx`

References

1. Aken, J.v., Andriessen, D.: Handboek ontwerpgericht wetenschappelijk onderzoek. Boom Lemma (2011), In Dutch
2. Apostel, L.: Towards the Formal Study of Models in the Non-Formal Sciences. Synthese **12**, 125–161 (1960)
3. Dietz, J.L.G.: Enterprise Ontology – Theory and Methodology. Springer, Heidelberg, Germany (2006)
4. Dietz, J.L.G., Hoogervorst, J.A.P., Albani, A., Aveiro, D., Babkin, E., Barjis, J., Caetano, A., Huysmans, P., Iijima, J., Kervel, S.J.H.v., Mulder, H., Op 't Land, M., Proper, H.A., Sanz, J., Terlouw, L., Tribolet, J.M., Verelst, J., Winter, R.: The discipline of enterprise engineering. International Journal Organisational Design and Engineering **3**(1), 86–114 (2013)
5. Dietz, J.L.G., Mulder, J.B.F.: Enterprise Ontology – A Human-Centric Approach to Understanding the Essence of Organisation. The Enterprise Engineering Series, Springer, Heidelberg, Germany (2020). https://doi.org/10.1007/978-3-030-38854-6
6. Dietz, J.L.G., Mulder, M.A.T.: Demo specification language 3.7 (2017)
7. Hofstede, A.H.M.t., Proper, H.A.: How to formalize it?: Formalization principles for information system development methods. Information and Software Technology **40**(10), 519–540 (October 1998)
8. Lankhorst, M.M., Torre, L.v.d., Proper, H.A., Arbab, F., Steen, M.W.A.: Viewpoints and visualisation. In: Enterprise Architecture at Work – Modelling, Communication and Analysis, pp. 171–214. The Enterprise Engineering Series, Springer, Heidelberg, Germany, 4th edn. (2017)
9. Mulder, M.A.T.: Validating the demo specification language. In: Enterprise Engineering Working Conference. pp. 131–143. Springer, Heidelberg, Germany (2018). https://doi.org/10.1007/978-3-030-06097-8_8
10. Mulder, M.A.T., Proper, H.: Towards Enterprise-Grade Tool Support for DEMO. In: Practice of Enterprise Modelling (PoEM) 2020. Springer (2020), Forthcoming
11. Ogden, C.K., Richards, I.A.: The Meaning of Meaning – A Study of the Influence of Language upon Thought and of the Science of Symbolism. Magdalene College, University of Cambridge, Oxford, United Kingdom (1923)
12. Recker, J.: Scientific research in information systems: a beginner's guide. Springer, Heidelberg, Germany (2012). https://doi.org/10.1007/978-3-642-30048-6
13. Wang, Y.: Transformation of DEMO models into exchangeable format. Master's thesis, Delft University of Technology, Delft, The Netherlands (2009)
14. Wang, Y., Albani, A., Barjis, J.: Transformation of DEMO metamodel into XML schema. In: EEWC 2011: Advances in Enterprise Engineering V. pp. 46–60. Springer, Heidelberg, Germany (2011)