Software and Systems Modeling
https://doi.org/10.1007/s10270-021-00911-z

SPECIAL SECTION PAPER

On Enterprise-Grade Tool Support for DEMO

Mark A. T. Mulder[1,2] · **Henderik A. Proper**[3,4]

Received: 12 March 2021 / Revised: 28 May 2021 / Accepted: 30 June 2021

Abstract
The Design and Engineering Methodology for Organisations (DEMO) is a core method within the discipline of enterprise engineering. It enables the creation of so-called *essential* models of enterprises. Such models are enterprise models that aim to focus on the organisational essence of an enterprise by leaving out (as much as possible) details of the socio-technical implementation. The organisational essence is then expressed primarily in terms of the actor roles involved, and the business transactions between these roles. The DEMO method has a firm theoretical foundation. At the same time, there is an increasing uptake of DEMO in practice. This also results in a need for enterprise-grade tool support for the use of the method. In this paper, we report on a study concerning the selection, configuration, and extension, of an enterprise-grade tool platform to support the use of DEMO in practice. The configuration of the selected tool framework to support DEMO modelling, provided general insights regarding the development of enterprise-grade tool support for (model-driven) methods such as DEMO, while also providing feedback on the consistency and completeness of the DEMO specification language; the specification language that accompanies the DEMO method.

Keywords Enterprise engineering · DEMO · Modelling tools

1 Introduction

The DEMO [6] method is one of the core methods (based on a theoretically founded method*ology*) within the discipline of enterprise engineering (EE) [7]. It focuses on the creation of so-called *essential* models of enterprises. The latter kind of models aim to capture the organisational essence of an enterprise by leaving out (as much as possible) details of the socio-technical implementation. The organisational essence is then expressed primarily in terms of the actor roles involved and the business transactions [40] (and ultimately in terms of speech acts [15]) between these actor roles. More specifically, 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 one or more diagrams and one or more cross-model tables.

The DEMO method has strong methodological, as well as theoretical, roots [6,7,40]. At the same time, there is an increasing uptake of DEMO in practice. The latter is illustrated by the active usage (and certification) community[1], reported cases concerning the use of DEMO [2,5] and [8, chapter 19], as well as integration with other mainstream enterprise modelling approaches such as ArchiMate [20,41] and BPMN [3,14,27].

The increased uptake of DEMO has also triggered the need for enterprise-grade tool support. Possibly due to its academic roots, there has not been much attention for the development of enterprise-grade tool support so far (we will discuss the notion of enterprise-grade in more detail in Sect. 3). As we will see in the remainder of this paper, tools supporting DEMO have indeed been developed. However, these generally involve either (advanced) academic prototypes, or are provided by smaller "boutique" tool vendors. These tools,

Communicated by Dominik Bork and Janis Grabis.

✉ Mark A. T. Mulder
markmulder@teec2.nl

✉ Henderik A. Proper
e.proper@acm.org

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, Belval, Luxembourg

[1] http://www.ee-institute.org/en
https://www.linkedin.com/company/enterprise-engineering-institute.

Published online: 26 July 2021

regretfully, do not classify as enterprise-grade tooling. Meanwhile, the lack of such tool support is actually hampering the further uptake of DEMO by larger organisations.

In this paper, which provides an elaboration on [33], we report on a study concerning the selection, configuration, and extension, of an enterprise-grade tool platform to support the use of DEMO in practice. The development of DEMO tool support also brought about the need:

1. for an elaboration of the DEMO meta-model to e.g. enable (automatic) verification and analysis,
2. to complement the core meta-model of the method with a visualisation oriented meta-model, and
3. to be able to exchange the models between tools, resulting in an exchange oriented meta-model.

This exercise also provided interesting insights into limitations of DEMOSL, the specification language that accompanies the DEMO method. The work, as reported in this paper, is actually part of a larger design science effort [32] to evolve two artefacts: (1) DEMOSL and (2) its associated tool support, towards a more enterprise-grade level.

When business analysts use DEMO in practice, i.e. to model complex organisations, modelling tools are an essential means to help create the needed models within an acceptable time frame, and with an acceptable level of quality. For example, tools also provide an important aid for modellers to maintain consistency between the different aspect models of a modelling method (such as DEMO). Without tools, or with separate and disconnected tools for a few aspect models, one is, e.g. not likely to spot incoherences between the different aspect models. Therefore, when developing such tool support, it is also important to ensure that the respective meta-models of all aspect models, as well as the associated diagrams and tables, are aligned in order to create an integrated and consistent meta-model. Automated tool support also requires an unambiguous logic and must also accommodate the different aspect models. Therefore, developing a tool that can support a method such as DEMO is not straightforward. Even more, developing a modelling tool based on a method's meta-model will also bring out possible limitations of the method and vice versa.

In assessing the validity of the two artefacts (DEMOSL and associated tool support), driving the different design science iterations [39], we did not limit ourselves to fictitious examples but specifically also included (see Sect. 10) real-world situations.

It should be noted that this paper is not concerned with the comparison of DEMO to other methods. Such comparisons have indeed been indeed made [3,5,14,20,21,27,41], and actually indicate a complementary relationship between DEMO and, e.g. e3Value [21], ArchiMate [20,41], and BPMN [3,14,27]. However, in this paper, we focus on tool support for DEMO *as is*.

The remainder of this paper is structured as follows. In Sect. 2, we briefly visit the research methodological background of the work reported on in this paper. In Sect. 3 we then continue by providing more background to the notion of "enterprise-grade". Sect. 4 provides on overview of the DEMO method. In Sect. 5, we then zoom in on the meta-model of DEMO as it should be supported by a modelling tool, as such, providing a first set of requirements for such tool support. Sect. 6 then continues by identifying additional requirements for enterprise-grade DEMO tooling. Based on these requirements, Sect. 7 briefly reports on the assessment of relevant tools and platforms, resulting in the selection of Sparx Enterprise Architect for further experimentation. Sect. 8 reports on the configuration of the latter platform to actually support DEMO modelling, while Sect. 9 summarises some of our experiences in doing so. The latter also includes a reflection on the extensions and refinements needed to the DEMOSL (the specification language that accompanies the DEMO method). Finally, before concluding, Sect. 10 reports on cases where the resulting tooling has been used in practice.

2 Research methodology

Design science, according to Wieringa [51], is the design and investigation of artefacts, within a context, that are designed to interact with this context, and should improve something in that context. An artefact, in this context, is an object that solves a problem by interaction with the context of that artefact. The same artefact may or may not solve another problem in another context by interaction with the latter context. Or in Wieringa's definition: *"An artefact is something created by people for some practical purpose. …artefacts, are designed, and these designs are documented in a specification."* [51].

Designing an artefact may result in multiple new artefacts that need to be designed. The resulting artefacts may answer knowledge questions. Thereafter, the artefact must be implemented, validated, and evaluated. The implementation, and hence the application of the artefact, takes place in the original problem context, thereby validating that this application benefits the stakeholder goals, if implemented. The final evaluation of the artefact, involves the evaluation of the results of the implementation with one or more stakeholders in the field. One can subsequently use these results to improve the artefact.

When designing, one typically investigates, creates and validates the design of the artefact(s) in an iterative cycle. This problem-solving process cycle is a logical structure to be able to engineer and design an artefact. The main task in this cycle is to understand the problem and choose the right design to resolve the problem before implementation.

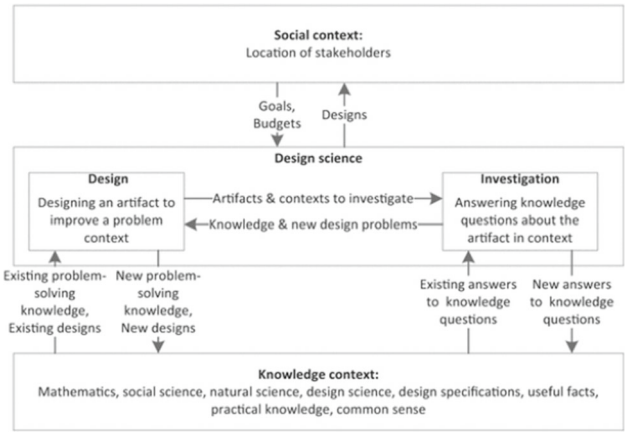

Fig. 1 The Wieringa [51] DSR, framework

Through evaluation, we can learn from this implementation towards the next cycle [51].

As mentioned before, in the research as reported on in this paper, we are concerned with two artefacts: (1) DEMOSL and (2) its associated tool support.

When we map our (primary) design science artefact, i.e. the DEMO tool, on the DSR (Design Science Research) framework by Wieringa [51], we see that the social context pertains to the DEMO practitioners that need the tooling to model organisations. The knowledge context is based on DEMO (see Sect. 4), DEMOSL, as well as knowledge regarding the Sparx Enterprise Architect platform. This base allowed us to create the first iteration of the tool. In every further iteration we made a design of modelling elements to be included, (see Sect. 5), in conjunction with the involved diagram of the DEMO methodology, and then created the elements and diagram within the tool (see Sect. 8). The tool was tested for its capabilities to model an organisation in various real life situations (see Sect. 10). Whenever we came to practical design problems we went back to the design stage to create a new iteration of the tool. In this design phase we used knowledge from the DEMO knowledge base but also other notation sources. In a few cases we created new DEMO related knowledge (see Sect. 9) and tested it in the tool environment within the social context of practitioners using the tool to model organisations.

3 Enterprise-grade tool support

The term "enterprise-grade" does not originate from science[2], but is a term commonly used in practice. As reported by practitioners, the term is used to differentiate consumer products from enterprise products [38]. In [11], Gartner defines it as: "*Enterprise-grade describes products that integrate into an infrastructure with a minimum of complexity and offer transparent proxy support.*" In line with this, enterprise-grade (in the context of software applications in general) is associated by practitioners [19,38,45] to characteristics such as *productivity*, *security* (including e.g. encryption of data, data security, granular levels of user access, protection of data, compliance), *integration*, *administration*, *support*, and *scalability*.

For a modelling tool to be enterprise-grade, it also implies that there must be a robust and secure repository (to store the models). Essential qualities for such repositories also include *extensibility*, *maintainability*, *interoperability*, and *portability* [35].

Since enterprise engineering and architecting efforts typically involve many different aspects, as well as many different stakeholders [23,37,49], it is important for modelling tools to provide different visualisations. Even more, these visualisations should be in a format that can be easily integrated into standard presentation, and text editing software.

Finally, as often also included in evaluations by e.g. Gartner [11], an enterprise-grade tool needs to be backed by a company or organisation that can provide reliable after-sales service and maintenance. This also presupposes a committed and serious tool vendor, while the tool needs to be built using mature technologies (which certainly does not automatically imply proprietary technologies).

In conclusion, "enterprise-grade" primarily concerns non-functional quality criteria of both the tool, and the technology vendor providing the tool and/or associated services.

4 DEMO

DEMO supports the modelling of the overall, as well as the more detailed, processes in an organisation, and the underlying information processing and structure. As mentioned before, the DEMO method has a strong methodological and theoretical foundation [6,7,40], while at the same time, there is an increased uptake of DEMO in practice [2,5,8]. This increased uptake in the practice of DEMO can be partially attributed to the complementary perspective that DEMO offers next to mainstream enterprise modelling languages such as ArchiMate and BPMN. Meanwhile, DEMO has gained a proven track record in process (re)design, software specifications based on the organisation, modelling business rules and proving compliance to, e.g. the General Data Protection Regulation (GDPR) [13] and other International Organisation for Standardisation (ISO) and NEderlandseNorm(NEN) norms [31]. Additionally, an early descendant of DEMO, called 'Voorwaarden scheppen voor

[2] Even though we can not claim to have conducted an in-depth literature survey on the term "enterprise-grade", the papers we did find (through a basic google scholar search) left the definition implicit.

de Invoering van Standaardisatie ICT in de bouw' (VISI), has evolved into the ISO 29481−2 : 2012[3] (BIM related) standard for the construction sector.

The DEMO method [6] identifies four key aspect models which we will summarise below. Each aspect model involves its own kinds of diagrams and tables, providing viewpoints on the complete model. The aspect models overlap in elements. Specifically, the concept of *transaction kind* appears in all aspect models; therefore, being the de-facto integration point of the model as a whole. A complete DEMO model covers the product, process, and information view on the essential organisations. These views are linked by the action rules that, in organisations, are often called business rules.

DEMO is built on a set of theories, where the Performance in Social Interaction theory (PSI-theory) constitute the foundation of enterprise ontology (EO) from a construction perspective. The PSI-theory divides an organisation into three worlds of acts and facts that reach commitments and agreements in predictable phases. One composes these agreements in a logical order that demands the right responsibilities of the right subjects. Finally, the subjects' capabilities are appointed into three categories to make distinctions in decisions and other actions. This standard pattern and production can be used anywhere in the organisation to enhance communication between subjects.

The PSI-theory identifies two worlds: communication (c-world) and products (p-world). Each of the c- and p-worlds may perform acts and produce facts. The communication acts, performed by actor-roles, produce communication facts. The same rule holds for the p-world. In short, actors perform communication and production acts, thereby creating communication facts about creating production facts. Furthermore, the PSI-theory defines the communication pattern between two actor roles. Every commitment to reach a production result follows this universal pattern, again expressed in the transaction kind.

Construction Model—The CM is the first and most comprehensive model to produce when modelling an organisation in DEMO, applying the Organisational Essence Revealing (OER) method. A CM is a model that represents the construction of an organisation. This model consists of the identified transaction kinds and the actor roles that are either executor or initiator. 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 the initiator of no, one or more transaction kinds.

The Construction Model (CM) involves the Organisation Construction Diagram (OCD) showing the Transaction Kind (TK), Aggregate Transaction Kind (ATK), Elementary Actor Role (EAR), Composite Actor Role (CAR) within a Scope of Interest (SoI). These diagrams show the dependencies between roles in execution and information. The high abstraction level makes this a compact diagram concerning the implementation of the organisation.

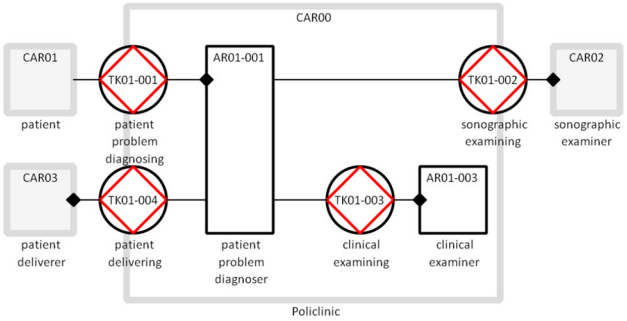

Fig. 2 Poligyn OCD

Name	Alias	Product Kind Formulation
TK01-004	patient delivering	the patient of [patient problem] is delivered
TK01-003	clinical examining	[clinical examination] is performed
TK01-002	sonographic examining	[sonographic examination] is completed
TK01-001	patient problem diagnosing	[patient problem] is diagnosed

Fig. 3 Poligyn TPT

The Transaction Product Table (TPT) shows the TK identification and description together with the product identification and description. This table is used to get insight into the products created by and for the organisation.

Finally, the Bank Contents Table (BCT) shows the contents of the ATK. This contains the identification and name of the ATK and the Entity Type (ET) and attributes of those ETs that are present. This is used to show the extend of (external) data. Figures 2 and 3 provide examples[4] of an OCD, and a TPT respectively.

Process Model—The PM bridges its CM and the coordination part of its AM. To this end, it specifies dependencies on how the transaction kinds in a tree are related to each other. More precisely, it specifies which transaction steps in an enclosed transaction kind are connected to which steps in the enclosing transaction kind, and by which kind of link (response-link or wait-link).

The Process Model (PM) involves a single diagram kind, the Process Structure Diagram (PSD), which shows the relations between the process steps of interrelated transactions.

[3] https://www.iso.org/obp/ui/#iso:std:iso:29481:-2:ed-1:v1:en.

[4] These examples are from the training case Poligyn of DEMO 3 training, also available in Ch.17 [8].

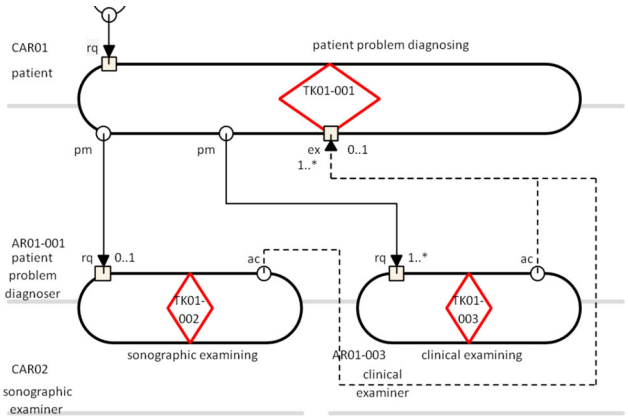

Fig. 4 Poligyn PSD

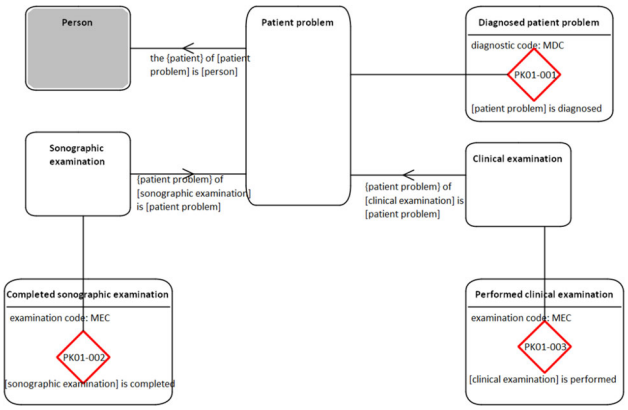

Fig. 5 Poligyn OFD

This is used to explain the order and dependencies between transactions. Business rules are partially covered as well. An example of a PSD is provided in Fig. 4.

Fact Model—The FM of a SoI bridges its CM and the production part of its AM. It specifies the various entity types, property types, attribute types, and entity types, and their mutual relationships.

The Fact Model (FM) also involves a single diagram kind, the Object Fact Diagram (OFD), which shows Entity Types (ETs) and Product Kinds (PKs), and a single table, the Information Use Table (IUT). This model is often called the data model, although it shows much more information.

Action Model—The AM comprises the guidelines that guide actors in doing their work, i.e. performing their coordination acts and their production acts. Action rules, which are actually (imperative) business rules, guide actors in responding to the coordination events they have to deal with. A semi-structured English-like language is used to express the action rules. Work instructions guide actors in performing the production acts, i.e. in bringing about the products of transactions.

This last aspect model contains Action Rules Specification (ARS) and Work Instruction Specification (WIS). Per non-trivial process step, at least one specification shows the input and conditions to proceed in the transaction pattern or advance to other transactions. This specification is used to model all details of the (to-be) organisation.

Different aspect models contain overlapping elements; therefore, the DEMO *essential model* results from the combination of all aspect models.

5 The DEMO meta-model

Earlier work [29] towards tool support for DEMO already resulted in improvements to the DEMOSL [9]. This includes, amongst others, the introduction of extra concepts in the ontological part of the meta-model, as well as updates to the verification rules, the data-meta-model and the exchange-meta-model. In this section, we discuss some of the additional extensions that have been made. Figure 6 provides a full overview of the resulting ontological-meta-model, while Fig. 7 shows the complete data-meta-model.

On the ontological level, common concepts have been added to the meta-model to support practical hierarchical concepts for actor roles as well as the concept of SoI. The final ontological meta-model (see Fig. 6) also incorporates the findings as reported in in [30]. In summary, the added relationships and entity type (shown in red in Fig. 6) are:

- appointing an EAR to an ATK to hide the EAR inside this composition.
- the hierarchical relation of CARs.
- the with clause of the ARS that connects the rules to the facts.
- the link between the TK and the facts of the ET that they produce.
- the aggregate relations between ET that provide the tree types of aggregation.

The combination of all DEMOSL's (partial) ontological meta-models, as well as the above discussed extra elements, results in the data-meta-model. This resulting data-meta-model is automation-oriented and includes all the properties that need to be captured about the different modelling concepts. However, it does not include the diagram-meta-model, which is concerned with the graphical layout of diagrams and tables. The data-meta-model, as created inside the tool implementation, does contain all meta structures for elements of the model, and connections of the model.

The data-meta-model also extends the original meta-model with attribute types that need to be registered as well as property types that are required in an implementation environment. This, in particular, involves property types that may

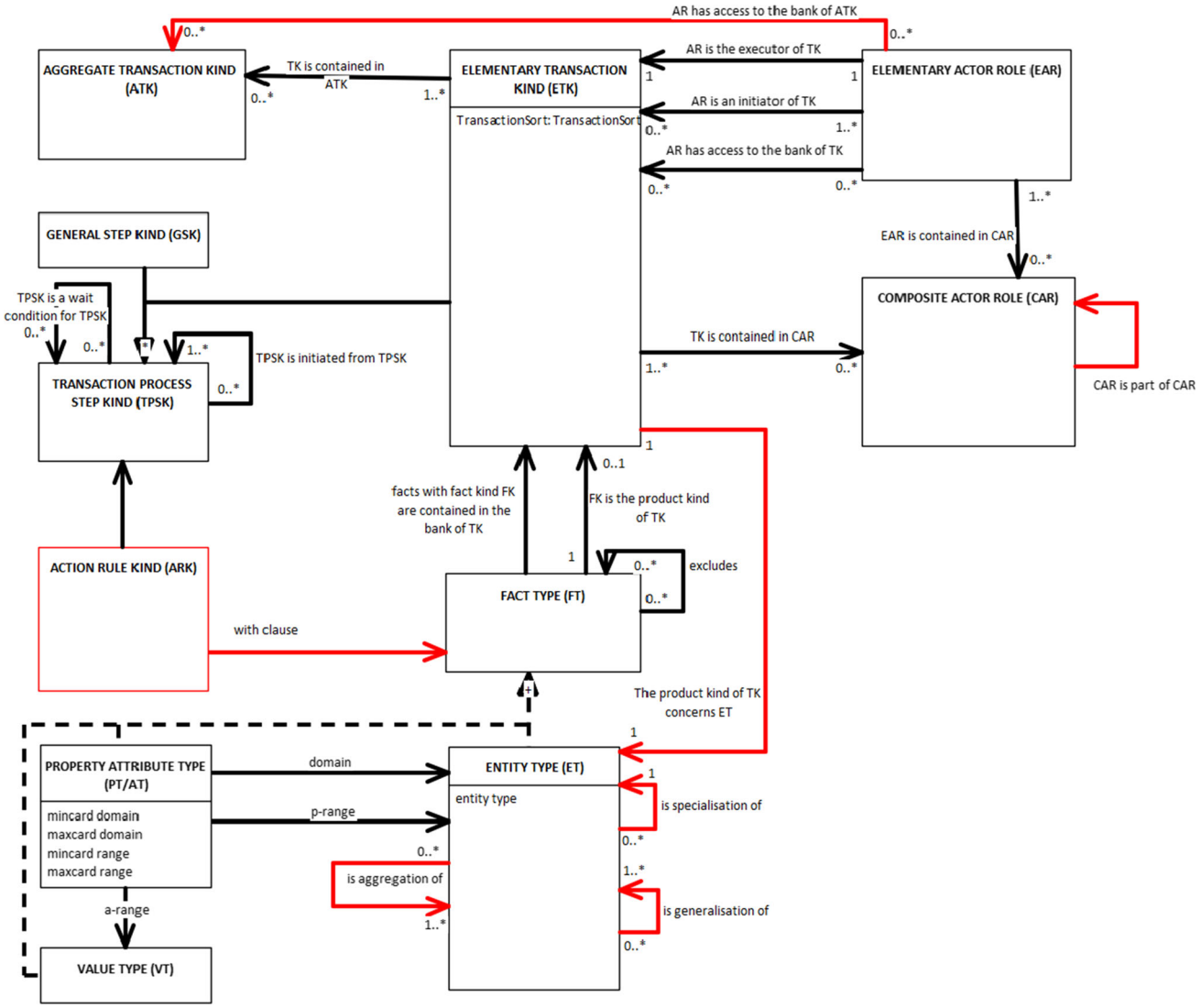

Fig. 6 DEMOSL ontological meta-model; red parts have been added from [29] to [9] and [30]

seem redundant in an ontological model but are necessary during the modelling process. The exchange-meta-model supports the storage and exchange of the allowed elements and connections. This model enables the interchangeability of DEMO models between possible modelling tools and other verification, simulation or translation tooling available or needed. The use of verification rules reduces the chance of creating incorrect DEMO models.

In general, 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 have defined:

1. Data-meta-model for the item and its property types.
2. Formal rules on the collections of items.
3. Exchange-meta-model of the objects:
   (a) XML Schema Definition (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
4. Programming model to implement the rules and data-meta-model.
5. Visualisation model in the selected tool platform.
6. Example model on the selected tool platform.

With this list of models and meta-models, the representation of the DEMO meta-model proved sufficiently complete [29] to validate the model and exchange information to recreate the model.

In contrast to the ontological meta-model, the data-meta-model does contain all implementation attribute types and

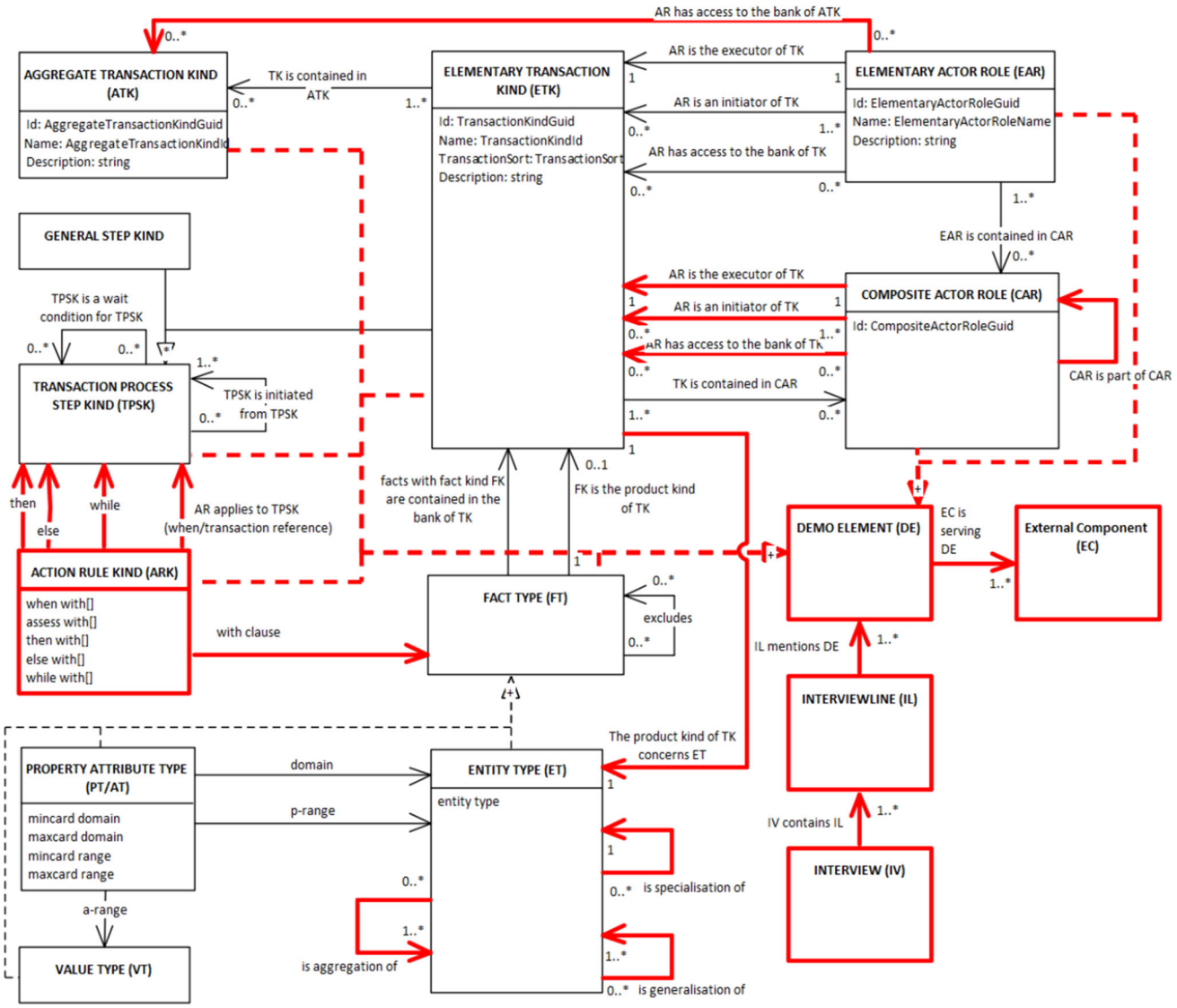

Fig. 7 DEMOSL data-meta-model; red parts have been added

property types for the DEMO meta-model. The data-meta-model of DEMO (see Fig. 7) has property types between the concepts. We did not visualise the removed property types, but these can be found by comparing our model with DEMOSL 3.7 (e.g. we removed a domain property type, precludes and precedes property types, Transaction Kind (TK) initiated from the Transaction Process Step Kind (TPSK)).

An example of the differences in the meta-models involves the Composite Actor Role (CAR). If a CAR is an initiator one should be able to represent it in the data-meta-model. Whenever one designs a CAR that initiates the transaction, this instantiation of this property type must be present in the model, and thus needs a representation in the meta-model. Therefore, the data-meta-model also has the property type '*AR is an initiator of TK*' from the CAR to the TK. In an ontological model, one could reason that behind every CAR an Elementary Actor Role (EAR) is present that covers the property type for the initiation.

Both meta-models of DEMO show the existing property types and concepts in black. The added property types and concepts, as discussed above, are shown in red. This ontological meta-model is the base for the data-meta-model. The ontological meta-model is the closest meta-model to the meta-model that lists all ontological principles of DEMO models. For example, the CAR cannot ontologically be an initiator of a transaction kind because an underlying elementary actor must be the initiator. Therefore this property type does not exist in the ontological meta-model.

In summary, the added relationships and entity types (shown in red in Fig. 7) are:

– In the implementation the relationships between CAR and TK have been made explicit as a copy of the relationship between EAR and TK.
– The relationships between ARS and the TPSK haven been instantiated following the grammar rules of ARSs.
– Interview and interview-line entities have been added to allow for source reference (see Sect. 9.
– The relationship between DEMO entities and other notations has been added to allow for inter-notation relationships.

Furthermore, mathematical rules have been designed to make sure only the correct property type instantiations can be present in the final data-meta-model as shown in Fig. 7.

6 Tool selection requirements

For our research, we selected an enterprise-grade tool (framework) to implement tool support for DEMO. In this section, we discuss the criteria we used in making this choice. We do not pretend that this set is complete or adequate for all users of a DEMO modelling tool. However, for the purposes of our research, we deemed these to be appropriate. An earlier version of these criteria, as reported in [32], was already used by other researchers [14] as well.

In the selection of the tool (framework), the following overall criteria were used:

1. The tool must be able to support the executability of the final model;
2. The tool must support the ways of modelling and thinking as described in the DEMO methodology;
3. The tool must support the interchange of models, in particular, conform the VISI[5] standard;
4. The tool must check the mathematical correctness of the model;
5. The tool must have efficient storage of the model;
6. The tool must support the validation of a model with the stakeholders.

From this list, we derived the more specific requirements as discussed below.

When using DEMO to model organisations, one should really be able to use all aspect models. This results in the following requirement:

Requirement 1 *The tool must support the creation of all four aspect models of DEMO.*

[5] The 'Voorwaarden scheppen voor de Invoering van Standaardisatie ICT in de bouw' (VISI) standard originates from the construction sector, where a considerable workflow automation group has developed a DEMO-based set of standards.

The specification of DEMO [9] provides the ontological meta-model that needs to be supported, as well as the visual representations of the different models and diagrams. Although the DEMOSL is not complete (yet), it certainly is the minimal description and guideline to visualise DEMO diagrams, as well as the minimum set to create a data-meta-model to describe the internal relationships of a DEMO model. To ensure compliance with these specifications, we need the following requirement:

Requirement 2 *The tool must be fully compliant with the DEMO meta-model, as specified in DEMOSL.*

One of the research topics in the field of DEMO is the integration and combination of other modelling methods and notations [22,24,27,41]. When we follow the recommendations of these publications, the best way to support the combination of modelling methods is to have a tool that supports various modelling methods and notations. This is such a practical benefit, apart from supporting further research, that we add it as a requirement:

Requirement 3 *The tool must support multiple modelling methods and notations.*

When going through DEMO models in published papers, we find incompleteness, structural incorrectness, and syntactic failures in various models [12,13,22]. To solve this problem, the model should be verified against the specification language. To conduct such a verification requires a considerable effort as it involves a large body of correctness rules. For example, the Xemod tool (see Fig. 7) already implements 23 rules for the CM [48]. When the task of verification takes more time than the modelling itself, we can accidentally introduce defects in the model. To compensate for the effort, the tool should automate the verification of the models as much as possible. This leads us to the requirement:

Requirement 4 *The tool must allow for model verification against the DEMO meta-model, DEMOSL.*

The DEMO methodology itself does not prescribe the modelling order of the model such that it is the only correct model [6]. The OER method only determines the components that need to be modelled and how these components must be connected. As a consequence, the model could not be built solely using Requirement 4. Therefore, we need an extra requirement that allows for variation in modelling order:

Requirement 5 *The tool must allow for model verification for any order in which a model is created.*

Although four meta-models were specified in the original DEMOSL, we created the integration of these four meta-models into a single, integrated meta-model. This single

meta-model allows for model verification and reuse of components of the model. We find the usage of a single repository useful, and therefore, we state the requirement:

Requirement 6 *The tool must store produced models in an integrated repository.*

Modelling tools have been around for several decades, and still, there is no common modelling tool that is the de-facto standard for DEMO. The tools that are used for modelling DEMO seem to be missing something or are not capable of modelling all the aspect models. We looked at a selection of used tools. Although there might be more tools that are in use or that are capable, we limited our research to this set of tools. We conjecture that existing tools are not optimal for DEMO in the standard available version. Therefore, we introduce our last requirement:

Requirement 7 *The tool must allow for extending the tool capabilities when this is deemed necessary for the full support of DEMO.*

A tool that can capture all DEMO related models and tables, and possibly even other related modelling techniques, may sound ideal. However, such a tool *also* needs to be viable in an enterprise context, which means that the enterprise-grade criteria as discussed in Sect. 3 become crucial.

Therefore, finally, considering the need to support the use of DEMO in larger enterprises, a selected tool needs to be *enterprise-grade*:

Requirement 8 *The tool (framework), and its provider, must be enterprise-grade (see Sect. 3).*

Finally, though usability of tools is an important subject, we have not included it in our considerations. On the one hand, usability from a user-interface point of view is highly subjective. Where one person might like mobile devices based on Apple interfaces, another person might likes Android interfaces, while both do the job just a bit different. On the other hand, in the context of modelling tools, usability is also strongly influenced by the notation [26] as used in a modelling language/method. Since we have taken DEMO as-is as our starting point, it would not be relevant (within our research) to consider the usability of the DEMO.

7 Considered tools

In searching for modelling tools that support modelling in DEMO and that comply with the requirements as listed above, we found eleven potential candidates. This initial selection was based on (1) existing experiences with tools within the community of DEMO practitioners, (2) potential or shown openness of the tool (framework) and/or tool vendor to accommodate DEMO models. The resulting ten candidates were then studied and checked in more detail against the requirements as discussed in Sect. 6.

We do not pretend to have conducted an exhaustive search for, and evaluation of, all available tools on the market. As discussed in the introduction, the lack of tooling was found to be a hampering factor in the further uptake of the DEMO method. A phenomenon, which we also observed to be the case in the first years in the existence of the ArchiMate [18] standard. Basically, a chicken-and-egg problem. For strategic reasons, we therefore primarily focused on finding a suitable tool among the enterprise (architecture) modelling tools that were already known-to, and/or used-by, members of the community of existing, and *potential*, DEMO users. The assumption underlying this strategy is that selecting an already used, or at least known, tool would also result in a quicker acceptance of a DEMO tool within the (initial) target audience. The latter will then, hopefully, also trigger additional (large) tool vendors to follow the example and include support for DEMO in their tool / framework as well. This is a pattern which we, think to also have observed in the uptake of the ArchiMate standard and associated tooling.

As such, building a tool from scratch was also not considered to be a desirable option, also when taking the enterprise-grade requirements into consideration.

In addition to the scores in terms of the requirements for DEMO tooling, we also came across some additional properties of tools that are worth mentioning. For instance, some of the considered tools have a cloud base, while others have a local running environment. Though these properties did not influence the choice it can be worth comparing in a later stage. Furthermore, some of the tools were able to simulate models. Because of our interest in simulation of models we added the information in the table. There are not a lot of tools available that are able to simulate DEMO models. During our search for relevant tools, some of the tools have actually been taken of the market. For reasons of completeness, we included this information in table 1 in terms of the *Simulation* and *Available* columns.

ARIS – by Software AG [44], is a business modelling tool. With the use of the ArchiMate modelling possibilities (Req. 3), the set of objects has been extended with a transaction and actor role to allow for Organisation Construction Diagram (OCD) modelling (Req. 1).

CaseWise Modeler – by CaseWise, is a business modelling tool that collects and documents the way in which organisations work. The tool has an Application Programming Interface (API) and can, therefore, be extended. Information about the extent of the API is unavailable [4]. The tool provides multiple diagrams, e.g. Entity Relation Diagram (ERD), Calendar and Process simulation (Req. 3). Its diagram features can be used to create DEMO diagrams except for the Action Model (AM), which is too complex for the tool

Table 1 Summary of the findings

	DEMO(1)	DEMOSL(2)	Multi-model()3)	Verifiable(4)	Business Rules(5)	Repository(6)	Extendable(7)	Enterprise-grade(8)	Total score	Cloud/Local	Simulation	Available
ARIS	C–	–	✓	–	–	–	–	✓	3	L	–	✓
CaseWise Modeler	CP-F	–	✓	–	–	✓	–	✓	4	L	–	✓
Connexio Knowledge System	C–	–	–	–	–	–	–	–	1	L	–	-
DemoWorld	C–	–	–	✓	✓	–	–	–	3	C	✓	✓
EC-Mod	CP-	–	–	✓	–	–	–		2	L	–	-
Enterprise Architect	––	–	✓	–	✓	✓	✓	✓	5	L	–	✓
ModelWorld	CP-F	–	–	–	✓	✓	–	–	3	C	✓	-
Open Modelling	CP-	–	✓	–	–	✓	–	–	3	C	–	✓
uRequire Studio	C–	–	–	–	–	–	–	–	1	C	–	✓
Visio	C-F	–	✓	–	–	–	✓	–	3	L	–	✓
Xemod	CP-F	–	–	✓	✓	✓	–	–	4	L	–	–

Legend for DEMO coverage: C = Construction Model, P = Process Model, A = Action Model, and F = Fact Model.
Total score = sum of ticks, plus 1 if at least the Construction Models are supported

(Req. 1). The repository can contain standard and extended objects for one model (Req. 6). Regretfully, no validation business rules can be added to the tool.

Connexio Knowledge System – by Business Fundamentals [43], documents the representation of the current organisation implementation using the Organisation Construction Diagram (OCD) diagram (Req. 1). It also stores Action Rules Specification (ARS) and Work Instruction Specification (WIS) as documents to enable people to do and understand their job knowledge management. It does not support all aspect models of DEMO. It is proprietary and for internal use only.

DemoWorld – by ForMetis [10], is an online modelling tool for DEMO processes (Req. 1). It does verify some business rules of the OCD (Req. 5) and allows for OCD modelling and process animation. The tool cannot model all aspect models and has no verification options.

EC-Mod – by Delta Change Consultants, can visualise organisational flaws in the transaction kinds and actor roles, using a modified OCD (Req. 1). This OCD shows organisation flaws and can be verified (Req. 4). It was presented in 2016 but is for internal use only.

ModelWorld – by EssMod [17], can support ArchiMate, DEMO, Business Process Model and Notation (BPMN), Unified Modelling Language (UML) and Mockups (Req. 3). The repository contains all models (Req. 6). It can visualise three aspect diagrams of DEMO (Req. 1), but these are not validated using business rules. Though the model can be simulated, no extensions can be written on this tool. Unfortunately, though this tool has been evaluated at the start of the research project, it is no longer available at the moment of writing this paper.

Open Modelling – [42], is a multi-model tool that supports Flowcharts, Integration Definition (IDEF) scheme, Application landscape, DEMO, ArchiMate, Use Case diagram, Component diagram, State diagram, ERD, Data Flow Diagram (DFD), and Screen sequence diagram.

uRequire Studio – by uSoft, is a requirement management tool [47] that has the option to add OCDs (Req. 1). The requirements repository cannot store DEMO models. Thus, diagrams are stored and only connected to the requirement definitions by object description. No other aspect diagrams of DEMO are available. The tool also supports BPMN diagrams (Req. 3) and is available for commercial use.

Visio – by Microsoft [25], is a drawing aid with templates for DEMO resembling the graphical representation (Req. 1). Multiple diagram types can be made in a single file (Req. 3). It does not have a repository of objects. Diagram validation is possible using programming in Visual Basic for Applications (VBA) (Req. 7), but diagram objects cannot easily be related.

Xemod – by Mprise, has the ability to integrally model three DEMO aspect diagrams within a single Scope of Interest per project file [48] (Req. 1), which might result in an inconsistency between multiple scopes of interests, scattered in models. Business rules can be used to verify the model on consistency between several elements (Req. 4, 5). Furthermore, the tool can show the OCD, PM, and FM as diagrams and lists (Req. 6). Unfortunately, this tool is no longer available on the market.

Finally, our tool of choice is **Enterprise Architect** – by Sparx, which is an analysis and design tool [46] for UML, SysML, BPMN, and several other techniques (Req. 3). Sparx Enterprise Architect (SEA) is broadly used, supports multiple models, can apply business rules, has a repository, is extendable and is already used for architecture modelling. It covers the process from analysis of requirements gathering through to model design, built, testing, and maintenance. Furthermore, it allows for the creation of meta-models to model one's objects, connections and business rules (Req. 5, 7). The repository stores all objects to enable reuse on multiple diagrams (Req. 6).

Table 1 provides a summary of the findings. As can be seen from the table, DEMOSL has not yet been implemented completely by any of the tools. More specifically, we conclude that no current tool can integrally capture all models and diagrams needed for method as a whole.

In principle, not having any tool that complies with all requirements, would have brought us to an impasse in the sense of tool builders waiting for a further increase of the use of DEMO, while potential DEMO users wait for enterprise-grade tool support. To move beyond this impasse, we decided to extend the commonly used framework of SEA.

The scores suggested that SEA would be the best candidate to fulfil the requirements, by using the extension framework of the tool for DEMO in either the Model Driven Generation (MDG) and/or the add-on functionality. More specifically, the SEA platform allows for the implementation of new/additional modelling languages by means of C# based extensions in combination with element definitions and "shape scripts" for the graphical shapes. Therefore, this tool framework was chosen to be configured with the DEMOSL meta-model and implement DEMO as precisely as possible.

8 Implementation

During our experiments, we used the SEA tool versions 13.0.1310 up to 14.1.1429. The modelling tool SEA has a meta-model base consisting of UML types. The tool allows for the extension of these basic UML meta-model and types with one's own "extended concepts". These extended concepts are stored in so-called profiles. By creating a new profile, these data types can be extended. Many meta-models in SEA are using these UML data types as a base for their own model (e.g. ArchiMate, BPMN). Therefore, the used

UML types for our meta-model are *Class* for entity types and *Association* for the relations between entity types.

To model the stereotype ≪*profile*≫ of the CM, we need to create an implementation of the DEMOSL concepts into the SEA modelling options. SEA uses a ≪*stereotype*≫ for each potentially visible object. Therefore, the meta-model of the CM needs to be reduced to visible entity types. In the meta-model of Construction Model (CM) of DEMOSL 3.7, we can see that the meta-model contains six entity types. The entity type Independent P-Fact Kind (IFK) or EVENT TYPE is the link between the Construction Model (CM) and the Fact Model (FM). It is not displayed on the Organisation Construction Diagram (OCD) and can be left out of the SEA meta-model for the CM, just like the FACT KIND. We already mentioned the missing Scope of Interest Boundary (SIB) in the CM meta-model. Therefore, we will use the Composite Actor Role (CAR) concept instead and remove IFK and reduce the number of ≪*stereotype*≫s concerning the CM to four, as shown in Fig. 8.

We extended the ≪*meta-class*≫ Class for the Elementary Actor Role (EAR), Composite Actor Role (CAR), Aggregate Transaction Kind (ATK), and Transaction Kind (TK) models, and added properties, attributes, and shape scripts to the stereotypes. Each stereotype has a "meta-type" property showing the default name of the instantiated model element. The how and why of other properties are available on request and will not be discussed in this section. We also extended the ≪*meta-class*≫ Association to create connectors between the elements.

The SEA allows for each element and link to be represented by a graphical shape (Fig. 10). The creation of the shape is supported by the programming language *shape script*. *Shape script* has a syntax similar to C but has a limited set of commands and functionality.

Within the shape scripts, it is possible to get the element properties or the containing diagram properties. See, for example, Fig. 9. These properties can be used in simple logic to determine the required shape to be drawn on the specific diagram. Limitations arise when shapes have different representations on different diagrams of the same type. When choosing the visualisation of a CAR, the programming features are too limited to either draw it as an internal rectangle and use it as Scope of Interest (SoI) or draw it as a grey-filled external rectangle automatically. The programming features do not allow for a context check of the visualisation. Therefore, these visual properties of the elements within SEA must be set manually at the moment resulting in all instances on all equal diagram types having the same appearance.

Next to implementing the actual modelling elements, and associated shape scripts for their visualisation, a model verification mechanism was implemented. Reasons for implementing these verification rules are that, first of all, the elements and connections can be added to the SEA repository by third parties using the API. The use of this API entry path will bypass the checking of the business rules because only User Interface (UI) events are available to be checked by the add-on in SEA. The presence of a verification engine allows modellers to create valid models. Secondly, the model may be entered to the best ability with all information available but is still incomplete. The verification emphasises the gaps and allows for corrective measures. Lastly, the tool has its limitations on keeping the diagrams within specifications

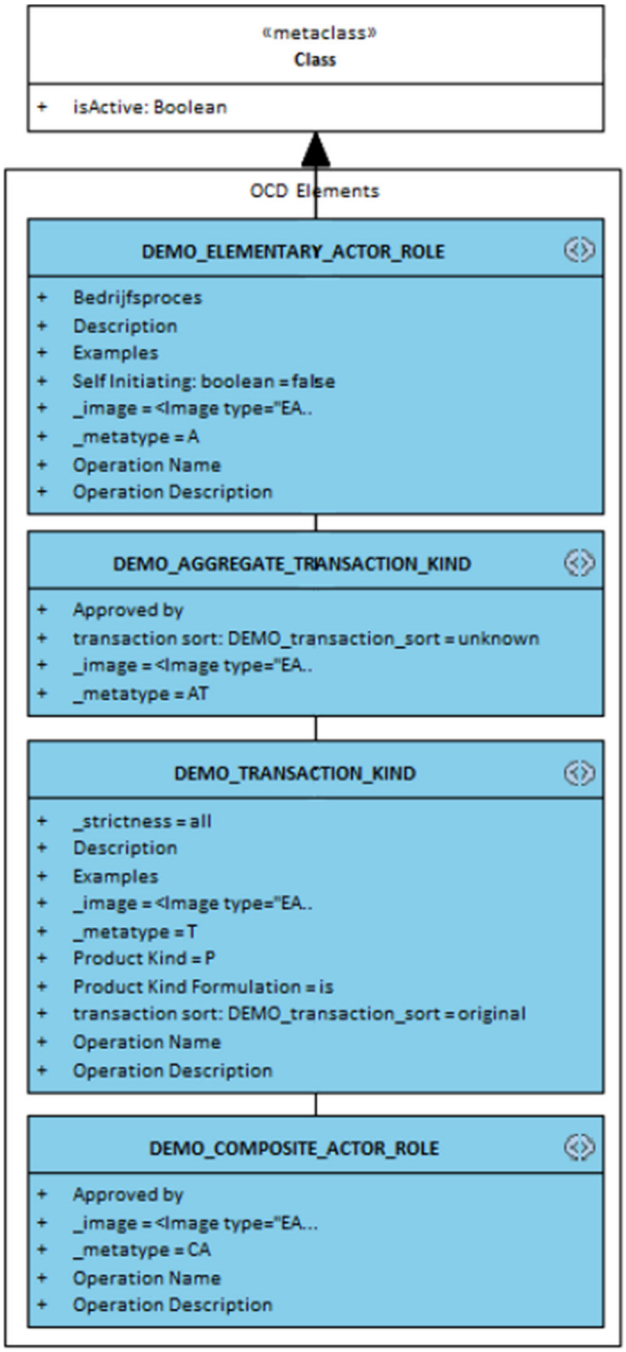

Fig. 8 OCD profile

```
shape OCDTransactionKind
{
   h_align = "center";
   v_align = "center";
   fixedAspectRatio = "true";
   setpenwidth(2);
   SetFillColor(255, 255, 255);
   SetPenColor(0, 0, 0);
   Ellipse(0, 0, 100, 100);
   setlinestyle("solid");

   if (HasTag("transaction sort", "original"))
   {
      SetPenColor(255, 0, 0);
   }
   else if (HasTag("transaction sort", "informational"))
   {
      SetPenColor(0, 255, 0);
   }
   else if (HasTag("transaction sort", "documental"))
   {
      SetPenColor(0, 0, 255);
   }
   polygon(50, 50, 4, 50, 0);
   print("#name#");
}
```

Fig. 9 Transaction Kind shape script

```
shape target{
    if (hasproperty("diagram.mdgtype", "DEMO::DEMO_PSD"))
    {
        MoveTo(0,0);

        SetFillColor(0,0,0);
        polygon(10,0,3,5,180);

        SetFillColor(252,242,227); // default color
        Rectangle(-5,-5,5,5);
    }
}
```

Fig. 10 Call Link shape script

```
public bool EA_OnPostNewElement(Repository repository,
            EventProperties info)
    {
      //Get element from the info
      int elementId = Convert.ToInt32(info.Get(0).Value);
      Element element =
        _repository.GetElementByID(elementId);
      ...
      Logger.Log("Created element {0}", ... );
      return false;
    }
```

Fig. 11 New Element Handling C#

of DEMOSL. Therefore, the diagrams can be corrected after new elements have been entered into the model.

The SEA tool has a flexible model extension feature but does not allow for a complete set of configurable business rules. Although some restrictions for inter-dependency can be made using, e.g. the Quick Link feature, most restrictions and checks have to be made using the API. This API allows for checking business rules at UI events such as menu click, element creation (see Fig. 11), deletion, and drag and drop onto a diagram canvas. We have used this API to implement business rules on relevant events.

The business rules that have been implemented for the Organisation Construction Diagram (OCD) are:

1. Elementary actor roles may only be the executor of a single elementary transaction kind
2. Every pair of actor role and transaction kind can only have a single connection of the type initiator, executor, or bank access
3. On an OCD diagram, only Aggregate Transaction Kind (ATK), Transaction Kind (TK), Elementary Actor Role (EAR), and Composite Actor Role (CAR) elements can be added.

All these rules have been codified into the verification model (see Fig. 12 part 6).

Finally, Fig. 12 shows both the tool "in action", as well as some details of the tool's *implementation* on the SEA platform. In the latter, we see (numbers referring to the respective windows as displayed in the figure) how SEA provides a set of base classes (1) for elements, connectors, diagrams, and toolboxes. Each DEMO model element involves an extension of one of the base classes of SEA (2), whole also having custom properties based on the DEMO meta-model (3). To enable various visualisations for different aspects, SEA provides simple scripts that enable the drawing of shapes (4). SEA provides interface "hooks", that are triggered when editing models in the graphical user interface, that can be used by add-on applications to provide functionality that is not available in SEA (5). The verification of DEMO models has been implemented in C# using these "hooks" (6).

9 Summary of experiences

In this section, we summarise some of our experiences in the development of tool support for DEMO. A more detailed account of these experiences is provided in [34]. This also involves the changes / refinement needed to the data-meta-model (resulting in the earlier discussed meta-model as shown in Fig. 7), as well as the need to add a meta-model dealing with visualisation, and with the exchange of DEMO models respectively.

A first interesting experience was the fact that, even though DEMO has a thorough theoretical basis, the original meta-model of DEMO was not specific and detailed enough to enable an immediate implementation. The latter can be explained by the fact that the book defining the DEMO method [6] was primarily written to teach learners (students and practitioners) to create DEMO models in accordance with the DEMO way of thinking, and draw (human to human) communicable DEMO models to reason about the organisation. As such, the book puts the priority on "doing", when introducing the different DEMO aspect model kinds. There was no need for a strict meta-model.

In line with this, the formalisation(s) provided in the original DEMO book [6] aimed to support didactic goals rather

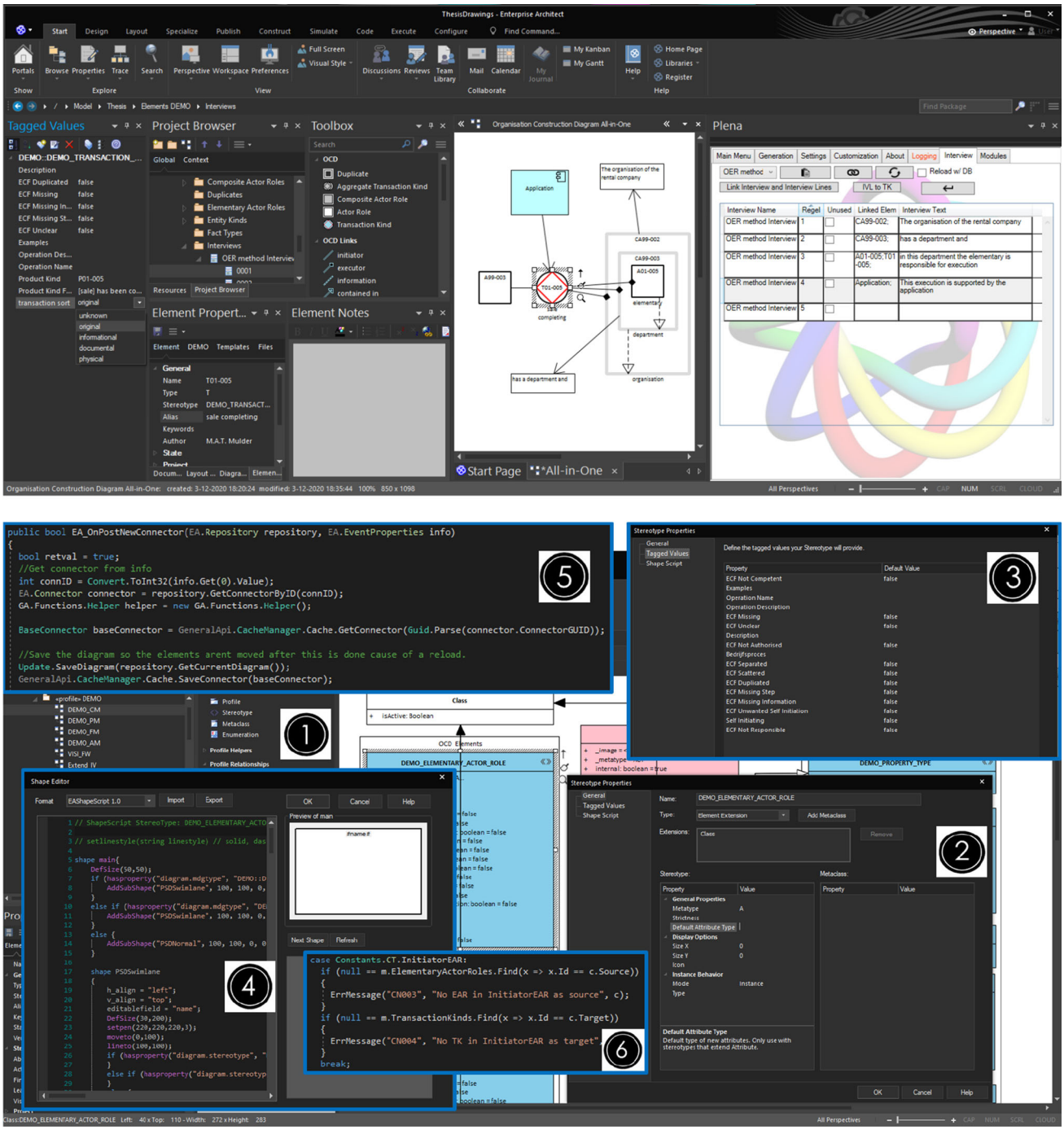

Fig. 12 Tool in action, and its implementation

than the development of automated modelling tools. As such, it was never meant to provide a formalisation and meta-models that would enable the development of, and automated support for, the methodology. As discussed in [16], different meta-modelling and formalisation goals will / should also result in formalisations with different level(s) of detail / specificity.

As a result, to enable tool development, a more complete and detailed formalisation and meta-model was needed. This triggered the initial development, and further evolution, of the DEMOSL [9], the specification language for DEMO. A first validation of the meta-model of this specification language was reported in [29]. As part of the (partial) validation [29], all existing DEMO (example) aspect models (taken from the

official course material and practical cases) were positioned within the specification language to see if they fitted.

As also discussed in Sect. 5, beyond the ability to "capture" actual DEMO models within the specification language, more extensions of the meta-model were needed to enable verification of the models as well as the operational use of the method in practice [30].

During the implementation, and practical validation, it also turned out that, for practical purposes, some of the concepts could actually be removed from the meta-model. For instance, according to the theory, each Organisation Construction Diagram (OCD) corresponds to an explicitly defined Scope of Interest (SoI). In practice, however, the SoI always corresponds directly to a Composite Actor Role (CAR). As a consequence, in practice, the SoI concept is redundant, and therefore we decided to not have it included in the meta-model explicitly.

A further interesting finding was the fact that DEMO allows modellers to start from any of the four aspect models. When learning the method, one generally starts with a Construction Model (CM) and gradually works down to the Action Model (AM); see Fig. 13. However, in practice, when interviewing domain experts in an organisation, these experts usually talk about the existing process and associated rules. In other words, starting with AM related information first. Additionally, the Fact Model (FM) information about entity types and attribute type is also provided relatively early when interviewing the domain experts.

The original DEMO meta-model did not allow for models to "grow" from the different aspect models, in the sense that the consistency rules would require the model to always be complete as a whole (so, including all aspect models). Therefore adjustments to the meta-model needed to be made to allow for such flexibility, while still enforcing (at the end of the modelling process) the overall consistency.

Furthermore, in practice, organisation models that resulted from the Organisational Essence Revealing (OER) analysis often raised questions regarding the *origins* of the included transaction kinds. More specifically, traceability from the elements in the organisation model to the original OER analysis was missing. To support this kind of cross-reference of the OER analysis, we have added the interview and interview line concept into the meta-model. By registering every aspect of the OER analysis as a connection from the interview line to the elements modelled from that line information we have created a traceable model from source to the final model. This interview notation technique is also illustrated in the NEN publication [31].

Another finding is that in practice, there was a need for DEMO models to be related to their existing or planned implementation. For instance, a "serving" connector was introduced that can be used to connect DEMO model elements to, e.g., application components in ArchiMate's Application layer. With this connector, one can, for instance, point to application components that implement the transaction kind or Transaction Process Step Kind (TPSK). Another example of the need to be able to include more of the implementation context involves the introduction of the actor (type) that aggregate actor roles. Such actors types correspond to job functions in the functional area of the model and, therefore, combine all competences of the elementary actor roles that are aggregated. This can be used for HR implementation information.

Finally, the action rule specification, as described in the DEMO method and associated specification language, consists of a semi formalised way to describe the communication actions that result in a decision on either actor role. The ontological meta-model of the action rule was composed of a single entity type that had its only connection to the TPSK it was based on. To start using the Action Rules Specification (ARS) in a more formal way, a more elaborate definition was needed. Therefore, based on earlier work [1], a grammar to represent action rules was created as well.

Next to such extensions (and simplifications) to the data-meta-model, we also found it necessary to add a visualisation, and an exchange-meta-model. Since the DEMOSL [9] primarily focuses on consistency and completeness of DEMO models from a "content" perspective, it does not include any specifics about the actual representation of these models in terms of diagrams, tables and other possible visualisations. As such, the DEMOSL does not provide guidelines regarding the concrete syntax of models in terms of, e.g., shapes and icons to be used. When examining a corpus of models produced across different cases, we found various variations in drawings of elements that each could be interpreted as the convention of those elements. In moving towards (standardised) tool support, this had to be remedied in terms of an explicit meta-model of the allowed visualisations.

As an example, consider the diagram provided in Fig. 14. In an OCD, the shape is normally drawn as a circle enclosing a diamond and this circle is stretched in the PSD with the diamond displayed at a seemingly random position within this "stretched circle". In Fig. 14, the name of the transaction appears below, above, or at the side-top of the transaction. Furthermore, the diamond-shape is positioned at a certain percentage from the left side, the stretching is a random length, while the swim lane usage is not consistent (i.e. 07 has the same initiator and executor but is visualised on top of two swim lanes).

This, finally, takes us to the need to exchange DEMO models between tools; see Fig. 15 for an example. As reported in earlier work [33], a number of tools[6] can model (partial) DEMO models, while some of these even have a built-in

[6] Created by e.g. Bakker&Spees, Technia, Future Insight, and Formetis.

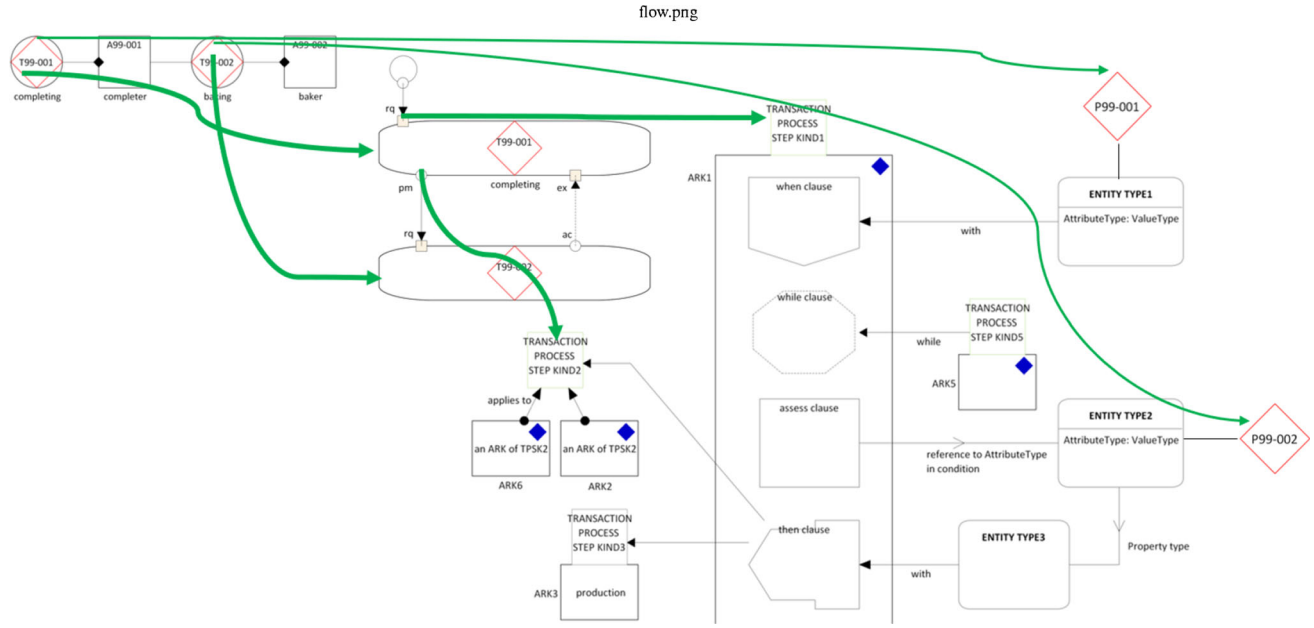

Fig. 13 CM-AM flow

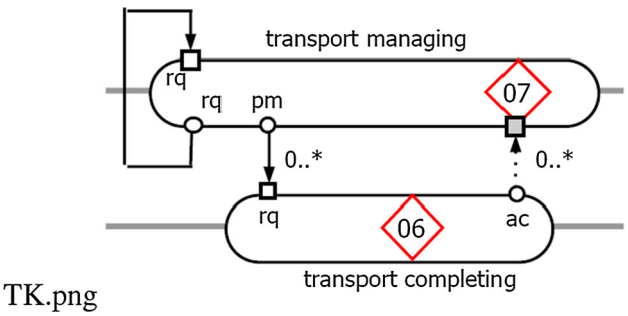

Fig. 14 PSD specification example

```
<?xml version="1.0" encoding="utf-8"?>
...
  <xs:element name="DEMOmodel">
    <xs:complexType>
      <xs:all>
        <xs:element name="TransactionKinds"
            minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="TransactionKind"
                type="TransactionKind"
                minOccurs="1" maxOccurs="unbounded">
...
  <xs:complexType name="TransactionKind">
    <xs:sequence>
      <xs:element name="Identification"
          type="TransactionKindId"></xs:element>
      <xs:element name="Name"
          type="TransactionKindName"/>
      <xs:element name="TransactionSort"
          type="TransactionSort"  default="unknown"/>
    </xs:sequence>
    <xs:attribute name="Id"
          type="TransactionKindGuid" use="required"/>
  </xs:complexType>
...
  <xs:complexType
      name="OrganisationConstructionDiagram">
    <xs:all>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Formulation" type="xs:string"/>
      <xs:element name="TransactionKindElements"
          minOccurs="0" maxOccurs="1">
...
```

Fig. 15 Part of exchange meta-model

engine to execute the actual models. It would be beneficial to be able to export and import DEMO models across tools because specific tools can have specific benefits when used in a certain organisational environment.

In addition, given the complementarity between modelling methods / languages, such as e3Value, BPMN, ArchiMate, and DEMO, it would be beneficial to also create conceptual bridges between the respective models. Experimental results regarding the potential benefits of this have been reported in e.g. [3,14,20,21,27,41].

To enable (interoperable) export and import of DEMO models across different tools and engineers, an exchange meta-model has been created that enables the exchange of both the actual model, including the different aspect models, as well as their visualisations. The exchange meta-model enables translation and connection to various other languages and includes model extensions to store models in other modelling languages such as ArchiMate and BPMN.

The exchange meta-model is based on XSD, due to its common availability in modern-day development environments [28,50].

A specific class of model exchanges needed in practice is the exchange between DEMO and VISI. As mentioned in Sect. 4, VISI is an early descendant of DEMO which has

now evolved into the ISO 29481 − 2 : 2012[3] (BIM related) standard for the construction sector. The exchange model between DEMO and VISI has been defined in the standard ISO 29481-2:2012.

In summary, the key experiences are:

- The enterprise ontology book [6] serves well for didactic purposes, but does not provide a formalisation of the method that can be used as a base for tool development.
- With regards to the actual DEMO method, the development of the tool, as well as the use of the tool in practical situation, resulted in:
  - The addition of extra concepts in the meta-models.
  - The addition of support for interviews in the context of the Organisational Essence Revealing (OER) diagrams.
- With regards to the DEMOSL:
  - More support was needed to deal with flexibility of modelling processes.
  - The DEMOSL was validated and had to be detailed further, in order to enable automation.
  - The latter also included the formalisation of the ARS diagrams.
  - A visualisation meta-model needed to be added to cater for visualisations.
  - Similarly, an exchange meta-model needed to be added to support the exchange of models.
- Finally, to support the connection to other enterprise modelling approaches, relationships needed to be added between DEMO(SL) and some of these other languages.

10 Cases

Over the past two years, we have created enterprise models for five organisations (as part of regular consultancy assignments), while using our tool for the creation and presentation of these models. The cases, which are labelled A-E, have been created for five companies in the Netherlands, across different domains [7] Case A, D, and E were used at logistic wholesale companies. Case B was used at a small property management company whereas case C regards a small call centre.

Case A involved a medical wholesale organisation which required only a construction model and a fact model of their organisation for the purpose of being aware of the organisation and communication structure. We modelled the organisational structure in DEMO and combined it with the overal architectural landscape expressed in terms of ArchiMate models. Though only Organisation Construction Diagrams (OCDs) have been made, the combination between the OCD of DEMO and ArchiMate's Application Layer Diagram (ALD) have been very useful, therefore confirming the need of Requirement 3. In this case the connections between DEMO and other notations have been identified.

Case B is a property management company that needed their processes and data modelled to be able to choose the right automation for their business. The complexity in this case was in the implementation. The OCD and OFD were the start of the implementation of both the landscape that was modelled in ArchiMate, as well as the application mapping. The OFD has been used for the base of the configuration of the domain application. The combination of the OFD and implemented entities is a concept that fulfills some demands and can be expanded. The security model that was needed for the application could neither be registered in DEMO nor in ArchiMate and needs further attention.

Case C is a small call centre that needed to choose a process and data matching application. We modelled the organisation and their landscape using DEMO (OCD and OFD) in combination with ArchiMate. By reverse engineering the software towards DEMO models we found a 80% match in process and data model. This was complete enough to buy the software licenses. Matching implementation and the organisation model is a part that is not completely covered by the tool in the used version. The meta-model lacks some connections between the various entity types.

Case D partains to a logistic wholesale company that is seeking for business optimisation. We mainly used the OCD to gather all products and processes in the organisation. With the help of the integration of the Disco process mining tool we gathered information at main TK and related that information to cash flow, processing time and material flows. The tool supported this modelling by combining process mining information, application landscape information (in terms of ArchiMate diagrams), as well as transaction and role information. Business optimisation was found in the analysis of the connections between the different departments [36].

Case E is a logistic wholesale company that lacked insight in the invoicing system. This analysis using just the OCD resulted in the insight that the communication was scattered around the organisation and the responsibilities were not defined. Implementation of a single actor role was among several subjects. The presentation of this information needs some improvement in the tool.

Finally, Table 2 provides an overview of the numbers of modelled elements per case.

[7] DEMO is a non-domain specific methodology which allows us to use the tool throughout the various domains.

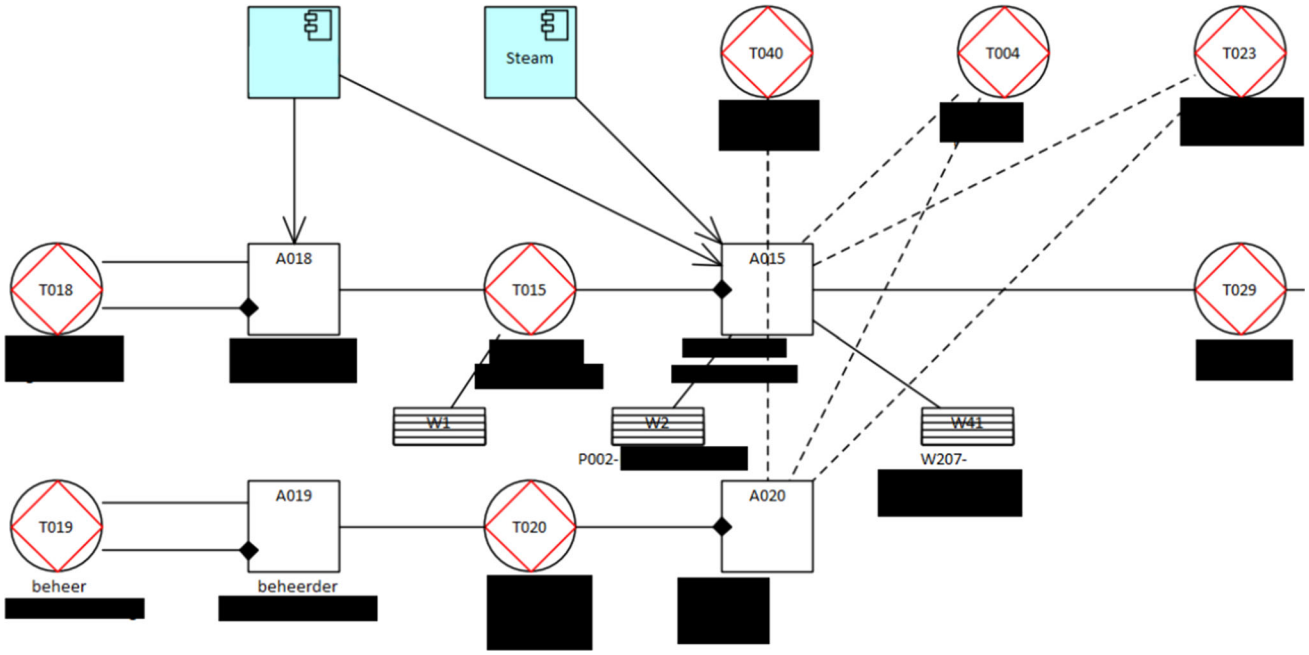

Fig. 16 Partial CM/OCD for case C; anonymised by blacking our specifics of names

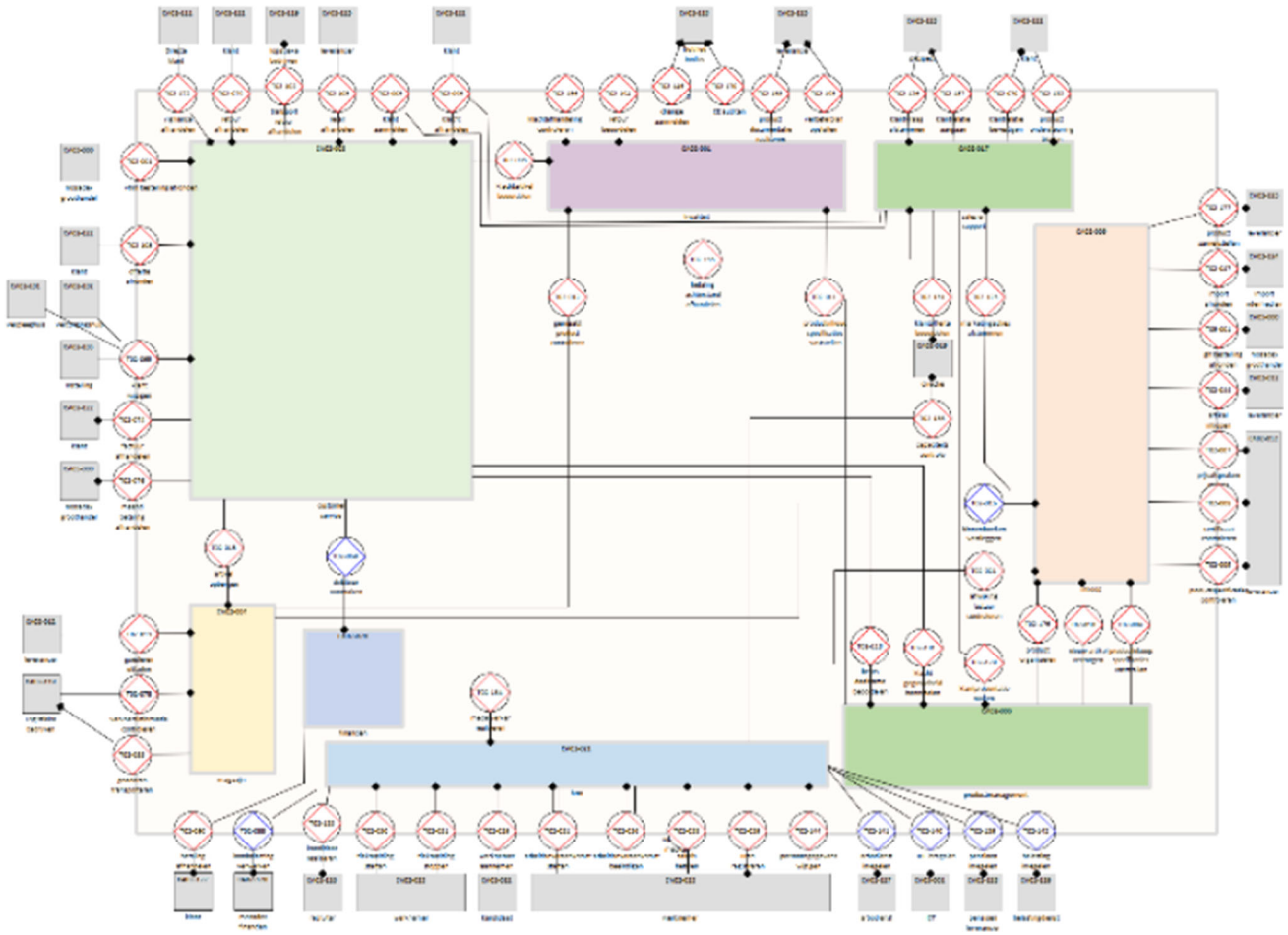

Fig. 17 High Level CM/OCD case D; anonymised by using a low image resolution

Table 2 Summary of the number of modelled elements per case

	Case A	Case B	Case C	Case D	Case E
Transaction kind (TK)	49	222	58	200	23
Elementary actor role (EAR)	38	191	21	168	20
Composite actor role (CAR)	12	36	19	81	10
Entity type (ET)	17	149	32	107	12
Action rules specification (ARS)	4	0	0	0	0
Work instruction specification (WIS)	0	9	55	3	0

11 Conclusions and future research

After two years of testing (in real-world cases) and optimising the tool we can conclude that, though not completely finished, the current version of the tool is capable of storing all DEMO models and visualise the business processes. The data meta-model that has been implemented in the tool appears to be rich enough to store the information of the mentioned cases. The base of the tool, SEA, is capable of supporting the modelling in DEMO. The resulting DEMO modelling tool has been used by and is freely available for researchers/lecturers in the academic world.

Even though the SEA platform allowed us to develop effective DEMO tool support, we also came across several limitations. For instance, SEA has not been built to display tables. Nevertheless, with some effort, we managed to still visualise tables. A more fundamental limitation pertains to the representation of graphical images. The visualisation engine of SEA is based on a 100 × 100-pixel image that is resized to the required dimensions. This allows for most graphical visualisations (squares) but falls short for independent resizable shapes (see Fig. 4). This same graphical concept is applied to connections between elements, which is also quite restricting.

Besides the discovered limitations of the chosen SEA tool framework, capturing the DEMOSL in SEA also revealed some missing definitions and inconsistent definitions in the DEMOSL. In future work, we plan to elaborate on these inconsistencies and try to make additions to DEMOSL to compensate or complete the specification.

Since a new version[8] of DEMO has been published in 2020 we need to investigate the gap between the current state of the research and the new information available. The newer version of DEMO also involves new diagram types and tables, which will have to be included in future versions of the tool. At first glance, some rules associated to the new diagrams and tables seem to complicate the modelling in SEA and might even go beyond its modelling capabilities. Therefore, next to the existing SEA environment, we are also starting research on a number of newly discovered tools/frameworks to investigate if these tools can also use the add-on that has been developed to extend the modelling capabilities towards DEMO.

Parallel to the development of the meta-model and the tool supporting the meta-model, some helpful functionality has been build into the tool that needs further research. For instance, the DEMO method allows one to work along a path following the OCD, Coordination Structure Diagram (CSD), Transaction Pattern Diagram (TPD) and PSD diagrams. The tool is able to generate the next diagram from a chosen element in the initial diagram as a start to create the elements needed in the next modelling step. These methodology helpers have been developed in practice and are very beneficial to the modelling speed and return on modelling effort. They do, however, require further research to provide the modellers with more support.

Now that a DEMO tool has been created on top an existing enterprise-grade platform, the tool itself also needs be made more enterprise-grade in the sense tutorials on the use of the tools, a service desk, etc, to support individuals (and organisations) to use the resulting tool.

Acknowledgements We would like to thank the anonymous reviewers from both the Software and Systems Modeling journal as well as the Practice of Enterprise Modelling (PoEM 2020) conference. Their feedback has resulted in many improvements to the original paper. In addition, we would like to thank the participants of PoEM 2020, for their participation in the on-line discussions regarding the original version of this paper. These discussions have also provided us with additional inspirations for improvements to this paper.

[8] https://demo.nl/download/demo-specification-language-4-6-1/?wpdmdl=842.

References

1. Andrade, M., Aveiro, D., Pinto, D.: Bridging ontology and implementation with a new DEMO action meta-model and engine. In: Enterprise Engineering Working Conference, pp. 66–82. Springer (2019)
2. Aveiro, D., D. Pinto, D.: A case study based new DEMO way of working and collaborative tooling. In: 2013 IEEE 15th Conference on Business Informatics, pp. 21–26. IEEE Computer Society Press, Los Alamitos, California (2013). https://doi.org/10.1109/CBI.2013.12
3. Caetano, A., Assis, A., Tribolet, J.: Using DEMO to analyse the consistency of business process models. In: Advances in Enterprise

Information Systems II, pp. 133–146. CRC Press (2012). https://doi.org/10.1201/b12295-17
4. CaseWise: The CaseWise Suite and CaseWise Modeler (2016). http://www.casewise.com/product/modeler/
5. Décosse, C., Molnar, W.A., Proper, H.A.: What does DEMO do? A qualitative analysis about DEMO in practice: Founders, modellers and beneficiaries. In: D. Aveiro, J.M. Tribolet, D. Gouveia (eds.) Proceedings of the 4th Enterprise Engineering Working Conference (EEWC 2014), Funchal, Madeira, Lecture Notes in Business Information Processing, vol. 174, pp. 16–30. Springer, Heidelberg, Germany (2014). https://doi.org/10.1007/978-3-319-06505-2_2
6. Dietz, J.L.G.: Enterprise Ontology - Theory and Methodology. Springer, Heidelberg (2006)
7. 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., 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. Int. J. Organ. Des. Eng. **3**(1), 86–114 (2013)
8. 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 (2020). https://doi.org/10.1007/978-3-030-38854-6
9. Dietz, J.L.G., Mulder, M.A.T.: DEMO Specification Language 3.7 (2017). https://www.eei-test.nl/mdocs-posts/demo-specification-language-3-7/
10. Formetis: Online modeling tool for process design and animation (2017). https://www.demoworld.nl/Portal/Home
11. Gartner: Information technology glossary. website (2020). https://www.gartner.com/en/information-technology/glossary/enterprise-grade
12. Gonçalves, A., Sousa, P., Zacarias, M.: Capturing activity diagrams from ontological model. Int. J. Res. Bus. Technol. **2**(3), 33–44 (2013). https://doi.org/10.17722/ijrbt.v2i3.57
13. Gouveia, D., Aveiro, D.: Modeling the system described by the EU General Data Protection Regulation with DEMO. In: Enterprise Engineering Working Conference, pp. 144–158. Springer (2018)
14. Gray, T., Bork, D., De Vries, M.: A new DEMO modelling tool that facilitates model transformations. In: Enterprise, Business-Process and Information Systems Modeling, pp. 359–374. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-49418-6_25
15. Habermas, J.: The Theory for Communicative Action: Reason and Rationalization of Society, vol. 1. Boston Beacon Press, Boston (1984)
16. Hofstede, A.H.M.t., Proper, H.A., : How to formalize it?: Formalization principles for information system development methods. Inf. Softw. Technol. **40**(10), 519–540 (1998). https://doi.org/10.1016/S0950-5849(98)00078-0
17. Hommes, B.J.: ModelWorld (2015). http://ModelWorld.nl
18. Iacob, M.E., Jonkers, H., Lankhorst, M.M., Proper, H.A.: ArchiMate 1.0 Specification. The Open Group (2009)
19. Kepes, B.: What does enterprise grade really mean? (2013). https://www.forbes.com/sites/benkepes/2013/12/18/what-does-enterprise-grade-really-mean
20. Kinderen, S.d., Gaaloul, K., Proper, H.A.: On transforming DEMO models to ArchiMate. In: I. Bider, T.A. Halpin, J. Krogstie, S. Nurcan, H.A. Proper, R. Schmidt, P. Soffer, S. Wrycza (eds.) Enterprise, Business-Process and Information Systems Modeling – 13th International Conference, BPMDS 2012, 17th International Conference, EMMSAD 2012, and 5th EuroSymposium, held at CAiSE 2012, Gdańsk, Poland, June 25-26, 2012. Proceedings, *Lecture Notes in Business Information Processing*, vol. 113, pp. 270–284. Springer, Heidelberg, Germany (2012). 10.1007/978-3-642-31072-0_19
21. Kinderen, S., Gaaloul, K., Proper, H.A.: Bridging value modelling to ArchiMate via transaction modelling. Softw. Syst. Model. **13**(3), 1043–1057 (2014). https://doi.org/10.1007/s10270-012-0299-z
22. Krouwel, M.R., Martin Op 't Land, M.: Combining demo and normalized systems for developing agile enterprise information systems. In: Advances in Enterprise Engineering V, pp. 31–45 (2011)
23. 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, The Enterprise Engineering Series, 4th edn., pp. 171–214. Springer, Heidelberg, Germany (2017). 10.1007/978-3-662-53933-0_8
24. Meertens, L.O., Iacob, M.E., Nieuwenhuis, L.J.M., van Sinderen, M.J., Jonkers, H., Quartel, D.: Mapping the Business Model Canvas to ArchiMate. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC 2012), Trento, Italy, pp. 1694–1701. ACM, New York, New York (2012). https://doi.org/10.1145/2245276.2232049
25. Microsoft: Visio (2020). https://www.microsoft.com/en-us/microsoft-365/visio/flowchart-software
26. Moody, D.L. (2009) The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Softw. Eng. **35**(6), 756–779. https://doi.org/10.1109/TSE.2009.67
27. Mráz, O., Náplava, P., Pergl, R., Skotnica, M.: Converting demo psi transaction pattern into BPMN: a complete method. In: Aveiro, D., Pergl, R., Guizzardi, G., Almeida, J.P., Magalhães, R., Lekkerkerk, H. (eds.) Advances in Enterprise Engineering XI, pp. 85–98. Springer, Cham (2017)
28. Mulder, M.A.T.: Towards a Complete Metamodel for DEMO CM. In: OTM Confederated International Conferences, pp. 97–106. Springer (2018)
29. 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
30. Mulder, M.A.T.: A design evaluation of an extension to the DEMO methodology. In: Advances in Enterprise Engineering XIII. pp. 55–65. Springer (2019)
31. Mulder, M.A.T.: NEN 7513 modelled in DEMO (2020). https://teec2.nl/wp-content/uploads/2021/05/NEN7513.pdf
32. Mulder, M.A.T.: Enabling the automatic verification and exchange of DEMO models. Ph.D. thesis, Radboud University, Nijmegen, the Netherlands (Forthcomming)
33. Mulder, M.A.T., Proper, H.A.: Towards enterprise-grade tool support for DEMO. In: J. Grabis, D. Bork (eds.) The Practice of Enterprise Modeling. PoEM 2020, Lecture Notes in Business Information Processing, vol. 400, pp. 90–105. Springer, Heidelberg, Germany, Riga, Latvia (2020). https://doi.org/10.1007/978-3-030-63479-7_7
34. Mulder, M.A.T., Proper, H.A.: On the Development of Enterprise-Grade Tool Support for the DEMO Method. In: Proceedings of the 33rd International Conference on Advanced Information Systems Engineering (CAiSE 2021), Melbourne, Australia, Lecture Notes in Computer Science. Springer, Heidelberg (2021). Forthcoming
35. Nunn, R.: What is "enterprise grade software" ? (2015). http://tractsystems.com/what-is-enterprise-grade-software/
36. Op 't Land, M., Dietz, J.L.G.: Enterprise ontology based splitting and contracting of organizations. In: L.M. Liebrock (ed.) Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC 2008), Fortaleza, Ceará, Brazil. ACM Press, New York (2008). https://doi.org/10.1145/1363686.1363815
37. Op't Land, M., Proper, H.A., Waage, M., Cloo, J., Steghuis, C.: Enterprise Architecture—Creating Value by Informed Governance. The Enterprise Engineering Series. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85232-2
38. Partridge, T.: What exactly does "enterprise-grade" mean? (2017). https://www.linkedin.com/pulse/what-exactly-does-enterprise-grade-mean-todd-partridge

39. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manag. Inf. Syst. **24**(3), 45–77 (2007)
40. Reijswoud, V.E.v., Mulder, J.B.F., Dietz, J.L.G., : Communicative action based business process and information modelling with DEMO. Inf. Syst. J. **9**(2), 117–138 (1999)
41. Roland, E., Dietz, J.L.G.: ArchiMate and DEMO—Mates to Date? In: A. Albani, J. Barjis, J.L.G. Dietz (eds.) Advances in Enterprise Engineering III, Lecture Notes in Business Information Processing, vol. 34, pp. 172–186. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01915-9_13
42. Santbrink, J.v.: Open Modeling (2020). http://open-modeling.sourceforge.net
43. Severien, T.: Business Fundamentals—Verbeteren vanuit de essentie (2016). http://www.businessfundamentals.nl/
44. Software AG: ARIS. http://www.softwareag.com/
45. Sparks, G.: What does enterprise grade mean? Website (2020). https://www.quora.com/What-does-Enterprise-Grade-mean
46. Sparx: Enterprise architect (2017). https://www.sparxsystems.eu/start/home/
47. uSoft: uRequire Studio (2016). http://www.usoft.com/software/urequire-studio
48. Vos, J.: Business modeling software focused on DEMO (2011). http://wiki.xemod.eu
49. Wagter, R., Proper, H.A.: Involving the right stakeholders—enterprise coherence governance. In: Proper, H.A., Winter, R., Aier, S., Kinderen, S.D. (eds.) Architectural Coordination of Enterprise Transformation, The Enterprise Engineering Series, chap. 10, pp. 99–110. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-319-69584-6
50. Wang, Y.: Transformation of DEMO models into exchangeable format. Master's thesis, Delft University of Technology, Delft, The Netherlands (2009)
51. Wieringa, R.J.: Design Science Methodology for Information Systems and Software Engineering. Springer, Berlin (2014)

Mark A. T. Mulder is the owner of The Enterprise Engineering Company (TEEC2). Mark has over 25 years of experience in the field of IT, including software development, business analysis, and enterprise modelling. Currently, he is also the chairperson of the Enterprise Engineering Institute. Mark is finalizing his PhD on the meta-model and tooling for DEMO.

Henderik A. Proper Erik for friends, is an FNR PEARL Laureate, and senior research manager within the Computer Science (ITIS) department of the Luxembourg Institute of Science and Technology (LIST) in Luxembourg. He is also an Adjunct Professor in Data and Knowledge Engineering at the University of Luxembourg. Erik has a mixed background, covering a variety of roles in both academia and industry. His core research drive is the development of theories that work, while his general research interest concerns the foundations and applications of domain modelling. Over the past 20 years, he has applied this research drive and general research interest toward the further development of the field of enterprise engineering and enterprise modelling, in particular.