Towards Enterprise-Grade Tool Support for DEMO

Mark A. T. Mulder[1,2] and Hend*erik* A. Proper[3,4]

[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. The Design and Engineering Methodology for Organisations (DEMO) method is a core method within the discipline of Enterprise Engineering (EE). It enables the creation of so-called *essential* models of organisations, which are enterprise models focusing on the organisational essence of an organisation, primarily in terms of the actor roles involved, and the business transactions between these actor roles. The DEMO method has a firm theoretical foundation. At the same time, there is an increasing uptake of DEMO in practice.
With the increased uptake of DEMO also comes a growing need for enterprise-grade tool support. In this paper, we therefore 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 selection process resulted in the selection of Sparx Enterprise Architect for further experimentation in terms of configuration towards DEMO. The configuration of this tool framework to support DEMO modelling, also provided feedback on the consistency and completeness of the DEMO Specification Language (DEMOSL), the specification language that accompanies the DEMO method.

Keywords: enterprise engineering · DEMO · modelling tools

1 Introduction

The Design and Engineering Methodology for Organisations (DEMO) [5] method is a core method (based on a theoretically founded method*ology*) within the discipline of Enterprise Engineering (EE) [6]. The DEMO method focuses on the creation of so-called *essential* models of organisations. The latter models capture the organisational essence of an organisation primarily in terms of the actor roles involved, as well as the business transactions [24] (and ultimately in terms of speech acts [13]) 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.

DEMO has strong methodological, and theoretical, roots [24, 5, 6]. At the same time, there is an increasing uptake of DEMO in practice. The latter is illustrated by

the active usage (and certification) community[5], reported cases concerning the use of DEMO [4, 1] and [7, chapter 19], as well as integration with other mainstream enterprise modelling approaches such as ArchiMate [15, 25] and BPMN [2, 19, 12].

The increased uptake of DEMO, also triggers the need for enterprise-grade tool support. In this paper, we report on a study into the selection, and configuration, of a (generic) tooling platform to support the use of DEMO in practice.

Configuring tool support for DEMO also requires an elaborate formalisation of DEMO's meta-model, which also enables the automatic verification of models. This exercise also provided interesting insights into limitations of DEMO Specification Language (DEMOSL), the specification language that accompanies the DEMO method.

The remainder of this paper is structured as follows. Section 2 provides more background to the DEMO method. In section 3, we discuss the meta-model of DEMO as it should be supported by a modelling tool, as such providing a first set of requirements for tool support. Section 4 then continues by identifying additional requirements for enterprise-grade DEMO tooling. Based on these requirements, section 5 then briefly reports on the assessment of relevant tools and platforms, resulting in the use of Sparx Enterprise Architect for further experimentation. Section 6 reports on the configuration of Sparx Enterprise Architect to actually support DEMO models, also providing important feedback on the DEMOSL, the specification language that accompanies the DEMO method. Finally, before concluding, section 7 reports on some experiences on the use of the resulting tooling in practice.

2 DEMO

DEMO supports the modelling of the overall, as well as the more detailed, processes in an organisation, and the underlying information processing. As mentioned before, the DEMO method has a strong methodological, and theoretical, foundation [24, 5, 6], while, at the same time, there is an increasing uptake of DEMO in practice [4, 1, 7]. Meanwhile, it has a proven track record in process (re)design and reorganisations, software specifications based on the organisation, modelling business rules and proving General Data Protection Regulation (GDPR) and other International Organisation for Standardisation (ISO) and NEderlandse Norm (NEN) norm compliance.

The DEMO method identifies four key aspect models. Each aspect model involves its own kinds of diagrams and tables, providing viewpoints on the complete model.
Construction Model – 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 shows the dependencies between roles in execution and information. The high abstraction level makes this a compact diagram in relation to the implementation of the organisation.

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 in the products that are being created in the organisation.

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

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.

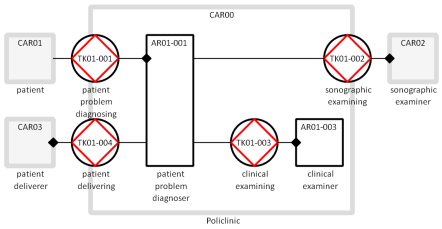

Fig. 1. Poligyn OCD modelled in the tool

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. 2. Poligyn TPT modelled in the tool

Process Model – The PM involves a single diagram kind, the Process Structure Diagram (PSD), which shows the relations between the process steps of interrelated transactions. This is used to explain the order and dependencies between transactions. Business rules are partially covered as well.

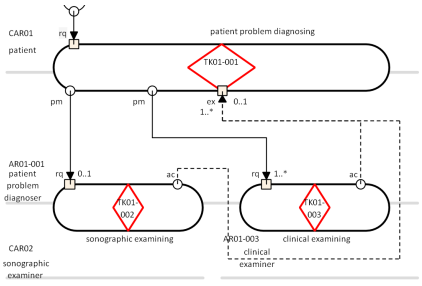

Fig. 3. Poligyn PSD modelled in the tool

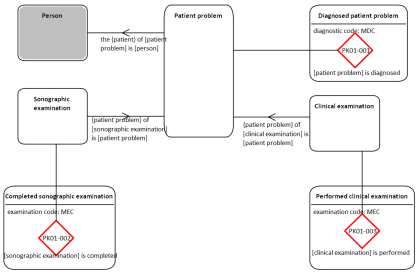

Fig. 4. Poligyn OFD modelled in the tool

Fact Model – The FM also involves a single diagram kind, the Object Fact Diagram (OFD), which shows ETs and Product Kinds (PKs), and the Information Use Table (IUT). This model is often called the data model although it shows much more information.
Action Model – The last aspect model is the AM with its Action Rules Specification (ARS). Per non-trivial process step minimal 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* is the result of the combination of all aspect models.

3 The DEMO Meta-Model

Earlier work [20] towards DEMO tool support for DEMO already resulted in improvements to DEMOSL 3.7 [8]. This has, amongst others, resulted in 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.

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 scope of interest. The data-meta-model extends the meta-model with attribute types that need to be registered as well as property types that that are required in a implementation environment. This, in particular, involves property types that may seem redundant in an ontological model but are necessary during the modelling process have been added. The exchange-meta-model supports the storage and exchange of the allowed elements and connections. This model promotes the interchangeability of DEMO models between possible modelling tools and other verification, simulation or translation tooling available or needed. The verification rules allow for reducing the chance of creating an incorrect DEMO model.

Combining all partial ontological meta-models from DEMOSL and extending this meta-model with the extra information, results in the data-meta-model.

This data-meta-model is automation-oriented includes all the properties we want to register for the concepts, but does not involve the diagram-meta-model which includes e.g. information about the graphical layout of diagrams.

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. Each structure of those elements and connections has some required and some optional properties. Instantiations of these element and connection structures make up the data model.

Subsequently, for every component of the meta-model we will define the data-meta-model, mathematical rules, exchange-meta-model (XML Schema Definition (XSD)), programming model, visualisation model and examples. For every diagram, there is a definition for the set of entity and property types called the diagram meta-model.

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

1. Data-meta-model for the item and its property types.
2. Mathematical rules on the collections of items.
3. Exchange-meta-model of the objects:
   (a) XSD specification for the set
   (b) XSD specification for the item
   (c) XSD specification(s) for the types
   (d) XSD specification for the diagram element
4. Programming model to implement the rules and 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 [20] sufficiently complete to validate the model and exchange information to recreate the model. We remodelled the meta-model according to the findings in [21].

The ontological meta-model of DEMO shows the existing property types and concepts in black. The new property types and concepts are added in red. This 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 contrast to the ontological meta-model, the data-meta-model does contain all implementation attribute types and property types for the DEMO meta-model. The data-meta-model of DEMO (see fig. 5) 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, TK initiated from the Transaction Process Step Kind (TPSK)).

The example mentioned above, about the CAR not being an initiator, is not valid in the data-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 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 EAR is present that covers the property type for the initiation.

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

4 Requirements for Tool Selection

For our research, we selected a tool (framework) to implement the DEMO meta-model based on the criteria as listed below. The criteria for selecting a tool are listed below. 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. Additionally, an earlier version of these criteria, as reported in [22], has also already been used by other researchers [12].

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 the conform the VISI[6] standard;
4. The tool must check the mathematical correctness of the model;
5. The tool must have an efficient storage of the model;
6. The tool must support the validation of a model with the stakeholders.

From this list, we derived the following requirements for the tool.

Modelling an organisation requires the use of all four DEMO aspect models. Therefore, we define the requirement:

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

[6] 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

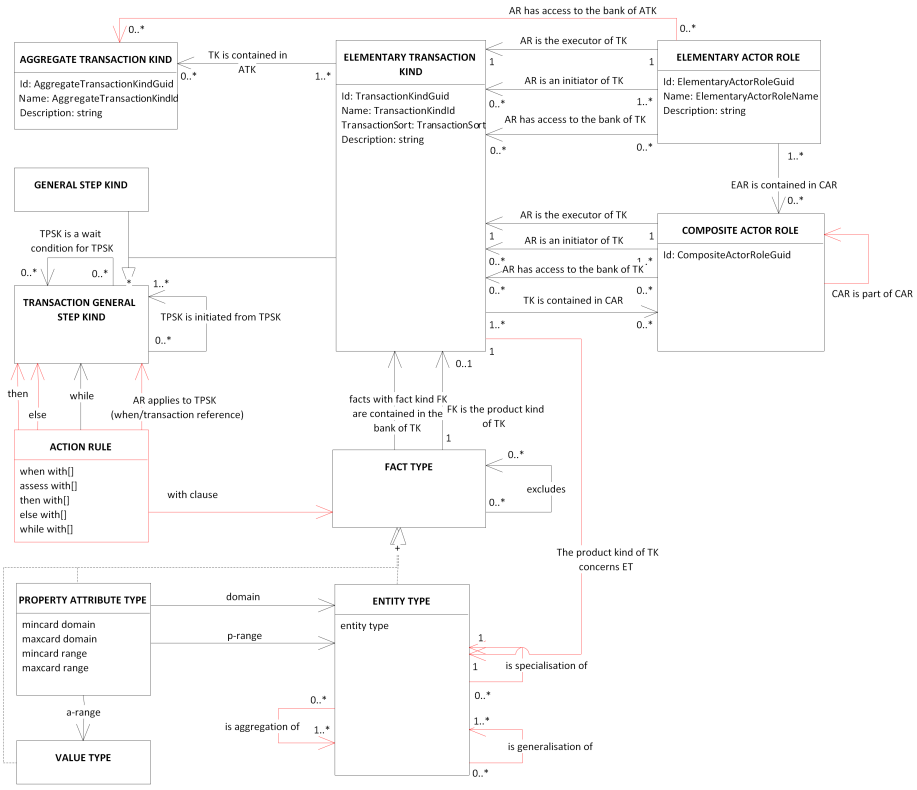

Fig. 5. DEMOSL data-meta-model

The specification of DEMO [8] states the meta-model and describes the visualisation. Although the DEMOSL is not complete it certainly is the minimal description and guideline to visualise DEMO diagrams and the minimum set to create a meta-model to describe the internal relationships of a DEMO model. To be compliant with these specifications, we need another 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 [25, 19, 16, 17]. 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 [16, 11, 10]. 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 Section 5) already implements 23 rules for the CM [31]. 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 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 [5]. The Organisational Essence Revealing (OER) method only determines the components that need to be modelled and how these components must be connected. Therefore, the model could not be build 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, integral 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 keep produced models in an integral 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 full support of DEMO.*

Finally, considering the need to support the use of DEMO in larger enterprises, a selected tool needs to be *enterprise-grade* in the sense that is must be backed by a company / organisation that can provide after-sales service and maintenance. This is not actually a requirement towards the actual tool, but rather on the provider. As such, the tools considered in this study (see section 5) were already pre-selected on this enterprise-grade requirement.

5 Considered Tools

In searching for modelling tools that support modelling in DEMO and that comply with the requirements as listed above, we have found ten candidates. In searching for possibly relevant tools, the enterprise-grade requirement was used as a pre-selection criterion. The resulting ten candidates have been studied and checked against the requirements.
ARIS – by Software AG [28] 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 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 [3]. 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 AM, which is too complex for the tool (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 [27] documents representation of the current organisation implementation using the OCD diagram (Req. 1). It also stores ARS and Work Instruction Specification (WIS) 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 [9] 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. It is available for commercial use. Students pay € 50 per 6 months, professors get free usage for 6 months.
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.
Enterprise Architect – by Sparx is an analysis and design tool [29] for Unified Modelling Language (UML), SysML, Business Process Model and Notation (BPMN), and several other techniques (Req. 3). 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).
ModelWorld – by EssMod [14] can support ArchiMate, DEMO, BPMN, 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, this tool is no longer available.
Open Modelling – [26] 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 [30] that has the option to add OCD (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 – [18] by Microsoft 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 integral model three DEMO aspect diagrams within a single Scope of Interest per project file [31] (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.

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

Table 1. Summary of the findings

Table 1 provides a summary of the findings. We conclude from it that DEMOSL has not been completely implemented in any tool yet. Moreover, we conclude that no current tool can integrally model the whole method. To get beyond this impasse of tool builders waiting for the usage and vice versa, we decided to extend a commonly used tool. The scores suggest that Sparx Enterprise Architect (SEA) is the best candidate to fulfil the requirements, by extending the tool with DEMO. SEA is broadly used, supports multiple models, can apply business rules, has a repository, is extendable and is already used for architecture modelling. Therefore, this tool has been chosen to model the meta-model and implement DEMO as precisely as possible.

6 Implementation

The modelling tool SEA allows for extending the basic UML model with the extenders own concepts. These extended concepts are called profiles. During this research, we used the tool version 13.0.1310 up to 14.1.1429. The modelling tool SEA has a meta-model base consisting of UML data types. 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 Construction Meta-model 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 CM and the FM. It is not displayed on the 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. We will use the CAR concept instead and remove IFK and reduce the number of ≪*stereotype*≫s concerning the CM to four, as shown in fig. 6.

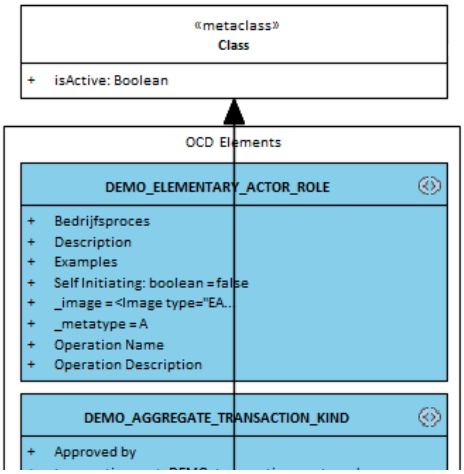

Fig. 6. OCD profile

We extended the ≪*meta-class*≫ Class for EAR, CAR, ATK, and TK 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 extended the ≪*meta-class*≫ Association to create connectors between the elements.

The SEA allows for each element link to be represented by a graphical shape. 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. These properties can be used in simple logic to determine the required shape to be drawn on the diagram. 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 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.

We implemented the verification of the model for multiple reasons. First of all, the elements and connections can be added to the SEA repository by third parties using the API. Using this API entry path will bypass the checking of the business rules because only User Interface (UI) events are available. The presence of a verification engine allows modellers to create valid models. Subsequently, 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. Furthermore, the tool has its limitations on keeping the diagrams within specifications 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, 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 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 ATK, TK, EAR, and CAR elements can be added.

All equations have been programmed info the verification model.

We need to verify the action model with the mathematical model and the exchange model. Therefore, we built five program parts of visualisation:

1. A verification model in code in the memory model (closely resembling the exchange model).
   This verification model is restricted to the action rule name, the TK and TPSK on which the rule applies, the entities involved in the with clause and, finally, the relations between the entities involved. For testing purposes, we used a simplified example.
2. A verbalisation generator to create the ARS from the memory model.
   This generator takes the memory structure tree and follows the structure depth-first towards the end of the action rule, thereby writing the language. The tree navigation is in compliance with the action rule grammar. More structures are allowed in the memory model than in the grammar due to the inherent strictness of the memory model.

3. A model converter to split the connected CM, PM and FM concepts needed for the AM.
   Although it would be more efficient to fetch the information directly from the original model, the assumed requirement of independence of the grammar module makes it necessary to create a new lookup structure for the model element.
4. An interpreter on top of the lexer-parser-listener of the created grammar that produces the exchange code.
   After lexing and parsing the input string, we can navigate through a tree which, again, follows the grammar structure. Now, instead of the language, we will fill the XML nodes into the XSD structure. Afterwards the nodes can be serialised to create an XML string.
5. A direct conversion from the memory model to the exchange model.
   The serialisation of the memory model can also be done via the XSD structure directly. This step creates an XML from the memory model.

At the end of the path, we have two XML files that have to be identical for the AM part. The AM must represent the same information in all formats.

7 Cases

Over the past two years, we have created business models for five organisations. We used our tool for the creation and presentation of these models. DEMO is a non-domain specific methodology which allows us to use the tool throughout the various domains. The cases, which are labelled A-E, have been created in five companies in the Netherlands. 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 is a medical wholesale organisation which needs a only construction model and a fact model of their organisation for the purpose of being aware of the organisation and communication structure. We modelled around 49 TKs, 38 EARs, 12 CARs, 17 ETs, 4 ARSs, 0 WISs. We modelled the organisational structure in DEMO and combined it with the landscape in ArchiMate. Though only OCD have been made, the combination between the OCD of DEMO and the ALD of ArchiMate 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. We modelled around 222 TKs, 191 EARs, 36 CARs, 149 ETs, 0 ARSs, 9 WISs. 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. We modelled around 58 TKs, 21 EARs, 19 CARs, 32 ETs, 0 ARSs, 55 WISs (see fig. 7). 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. Matchin 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.

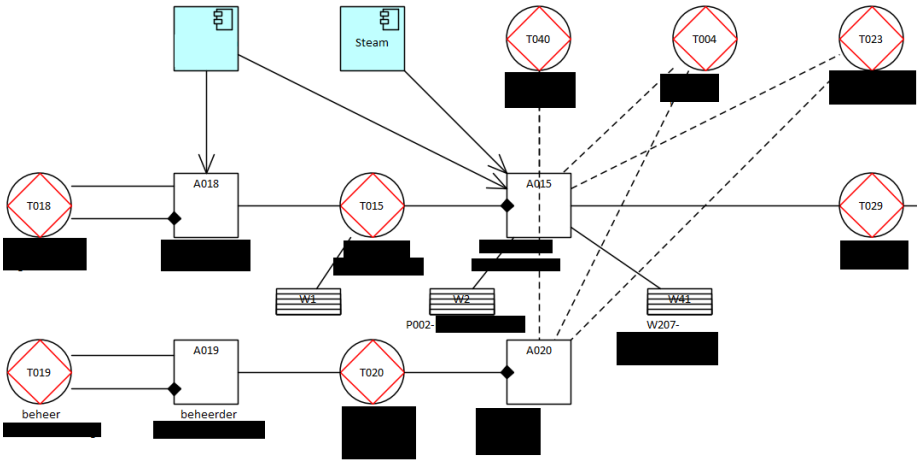

Fig. 7. Partial CM/OCD case C

Case D is a logistic wholesale company that is seeking for business optimisation. We mainly used the OCD to gather all products and processes in the organisation. We modelled around 200 TKs, 168 EARs, 81 CARs, 107 ETs, 0 ARS, 3 WISs. 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 proces mining information, ArchiMate landscape information and transaction and role information. Business optimisation was found in the analysis of the connections between the different departments [23].

Case E is a logistic wholesale company that lacked insight in the invoicing system. We modelled around 23 TKs, 20 EARs, 10 CARs, 12 ETs, 0 ARSs, 0 WISs. 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.

8 Conclusions and Future Research

After two years of testing and optimizing the tool we can conclude that, though not completely finished, the tool is capable of storing all DEMO models and visualise the business processes. The data meta-model that we implemented in the tool appears to be

sufficient to store the information of the mentioned cases. The base of the tool, SEA, is capable of supporting the modelling in DEMO 3. Although SEA has not been build to display tables, we have been able to visualise tables within the add-on.

The visualisation engine of SEA is based on a 100x100 pixel image that is resized to the required dimensions. This allows for most graphical visualisations (squares) but fall short for independent resizable shapes (see fig. 3). This same graphical concept is applied to connections between elements and is also quite restricting. Business rules have been implemented in the add-on and the freedom within this add-on compensates the lack of possibilities of the internal tool framework.

Besides the tool discoveries, modelling DEMOSL in SEA revealed some missing definitions and inconsistent definitions in DEMOSL. These will be discussed in another paper. More research is needed on the compliance of DEMO 4 and DEMOSL 4. Some rules of new diagrams and tables seem to complicate the modelling in SEA. We are also starting research on a number of newly found tools to investigate if these tools can also use the add-on that has been developed to extend the modelling capabilities towards DEMO.

References

1. 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)
2. 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 (Jun 2012)
3. CaseWise: The CaseWise Suite and CaseWise Modeler (2016), `http://www.casewise.com/product/modeler/`
4. Décosse, C., Molnar, W.A., Proper, H.A.: What does DEMO do? A qualitative analysis about DEMO in practice: Founders, modellers and beneficiaries. In: Aveiro, D., Tribolet, J.M., Gouveia, D. (eds.) Proceedings of the 4th Enterprise Engineering Working Conference (EEWC 2014), Funchal, Madeira. LNBIP, vol. 174, pp. 16–30. Springer (2014)
5. Dietz, J.L.G.: Enterprise Ontology – Theory and Methodology. Springer (2006)
6. 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)
7. 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 (2020)
8. 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/`
9. Formetis: Online modeling tool for process design and animation (2017), `https://www.demoworld.nl/Portal/Home`
10. Gonçalves, A., Sousa, P., Zacarias, M.: Capturing activity diagrams from ontological model. International Journal of Research in Business and Technology **2**(3), 33–44 (2013)
11. 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)

12. 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 (2020)
13. Habermas, J.: The Theory for Communicative Action: Reason and Rationalization of Society, vol. 1. Boston Beacon Press, Boston, Massachusetts (1984)
14. Hommes, B.J.: ModelWorld (2015), http://ModelWorld.nl
15. de Kinderen, S., Gaaloul, K., Proper, H.A.: On transforming DEMO models to ArchiMate. In: Bider, I., Halpin, T.A., Krogstie, J., Nurcan, S., Proper, H.A., Schmidt, R., Soffer, P., Wrycza, S. (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. LNBIP, vol. 113, pp. 270–284. Springer, Gdańsk, Poland (June 2012)
16. 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)
17. 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)
18. Microsoft: Visio (2020), https://www.microsoft.com/en-us/microsoft-365/visio/flowchart-software
19. 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 International Publishing, Cham (2017)
20. Mulder, M.A.T.: Validating the demo specification language. In: Enterprise Engineering Working Conference. pp. 131–143. Springer (2018)
21. Mulder, M.A.T.: A design evaluation of an extension to the DEMO methodology, pp. 55–65. Advances in Enterprise Engineering XIII, Springer (2019)
22. Mulder, M.: Enabling the automatic verification and exchange of DEMO models. Ph.D. thesis, Radboud University, Nijmegen, the Netherlands (Forthcomming)
23. Op 't Land, M., Dietz, J.L.G.: Enterprise ontology based splitting and contracting of organizations. In: Liebrock, L.M. (ed.) Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC 2008), Fortaleza, Ceará, Brazil. ACM Press (2008)
24. van Reijswoud, V.E., Mulder, J.B.F., Dietz, J.L.G.: Communicative Action Based Business Process and Information Modelling with DEMO. The Information Systems Journal **9**(2), 117–138 (1999)
25. Roland, E., Dietz, J.L.G.: ArchiMate and DEMO – Mates to Date? In: Albani, A., Barjis, J., Dietz, J.L.G. (eds.) Advances in Enterprise Engineering III, LNBIP, vol. 34, pp. 172–186. Springer (2009)
26. Santbrink, J.v.: Open Modeling (2020), http://open-modeling.sourceforge.net
27. Severien, T.: Business Fundamentals – Verbeteren vanuit de essentie (2016), http://www.businessfundamentals.nl/
28. Software AG: ARIS, http://www.softwareag.com/
29. Sparx: Enterprise architect (2017), https://www.sparxsystems.eu/start/home/
30. uSoft: uRequire Studio (2016), http://www.usoft.com/software/urequire-studio
31. Vos, J.: Business modeling software focused on DEMO (2011), http://wiki.xemod.eu