

A DEMO MODELLING TOOL THAT FACILITATES SEMI-AUTOMATIC DEMO-TO-BPMN TRANSFORMATIONS

T.J. Gray

Given the increasing emphasis on multi-perspective modelling, the University of Pretoria's Industrial Engineering Department requires a tool that enables horizontal transformation between DEMO (Design and Engineering Methodology for Organisations) models and other models. This tool needs to be available free-of-charge, up-to-date, and vertically consistent, and should enable horizontal transformation between DEMO models and other models such as the BPMN (Business Process Model and Notation) models. Using design science research, this dissertation develops a part of this tool by first identifying and evaluating currently available tools, then discovering how other authors have approached model consistency and transformation, and finally developing a tool in ADOxx that supports the modelling of DEMO's construction model— more specifically, the organisation construction diagram (OCD) and transaction product table (TPT) and transformations from the OCD to BPMN collaboration diagrams. The new tool, called DMT (Demo Modelling Tool), is demonstrated by modelling two case studies within the tool, and then performing four transformations. It is evaluated by having postgraduate participants model the same case study and rate the tool using pre-established usability criteria.

Keywords: Enterprise engineering; enterprise architecture; DEMO; BPMN; model transformation; conceptual modelling; meta-model

TABLE OF CONTENTS

Chapter 1	Introduction	5
1.1	<i>Problem context</i>	7
1.2	<i>Problem statement</i>	8
1.3	<i>Research questions</i>	8
1.3.1	Primary research question	8
1.3.2	Secondary research questions	8
1.4	<i>Dissertation statement</i>	9
1.5	<i>Scope demarcation, limitations, assumptions</i>	9
1.6	<i>Definition of terms</i>	10
1.7	<i>Significance</i>	11
1.8	<i>Document structure</i>	11
Chapter 2	Literature review	12
1.9	<i>Class-of-problem validation and solution areas</i>	12
1.9.1	Research method for the systematic literature review	12
1.9.2	Previous SLR results on class-of-problems	16
1.9.3	Previous SLR results on multi-view modelling solutions	17
1.9.4	SLR results on DEMO class-of-problems	19
1.9.5	SLR results on DEMO-related solution areas	22
1.9.6	Discussion	25
1.10	<i>Theory on DEMO aspect models</i>	28
1.10.1	The PSI theory	28
1.10.2	The ALPHA theory	30
1.10.3	The essential model of an enterprise	32
1.11	<i>Theory on BPMN</i>	41
1.12	<i>DEMO-BPMN transformation theory</i>	42
Chapter 3	Research methodology	44
1.13	<i>Literature on research methodology</i>	44
1.14	<i>Research methodology for the study</i>	44
1.15	<i>Expected rigour</i>	46
1.16	<i>Ethical considerations</i>	46
Chapter 4	Problem analysis and requirements analysis	48
1.17	<i>Data-gathering strategy and diagnosis techniques</i>	48
1.18	<i>Results for validating problem instance</i>	49

1.19	<i>Stakeholders or users that need a solution</i>	56
1.20	<i>Requirements analysis</i>	57
Chapter 5	Design and development	58
1.21	<i>ADOxx and the CM</i>	58
1.21.1	<i>ADOxx platform</i>	58
1.21.2	<i>ADOxx meta-model</i>	60
1.21.3	<i>CM meta-model</i>	62
1.21.4	<i>CM within ADOxx meta-model</i>	63
1.21.5	<i>CM meta-model compliance</i>	65
1.21.6	<i>TPT creation from OCD</i>	65
1.22	<i>DEMO to BPMN transformation</i>	66
1.22.1	<i>Scenario identification</i>	67
1.22.2	<i>Handling of user input</i>	68
1.22.3	<i>BPMN transformation</i>	70
Chapter 6	DMT demonstration	73
1.23	<i>CM demonstration for the college case</i>	73
1.24	<i>OCD to BPMN transformations for the college case</i>	74
1.24.1	<i>Scenario 1 transformation for college</i>	75
1.24.2	<i>Scenario 2 transformation for college</i>	77
1.24.3	<i>Scenario 3 transformation for college</i>	78
1.24.4	<i>Scenario 4 transformation for college</i>	79
1.25	<i>CM demonstration for the Rent-a-Car case</i>	81
1.26	<i>OCD to BPMN transformations for Rent-a-Car</i>	83
1.26.1	<i>Scenario 1 transformation for Rent-a-Car</i>	83
1.26.2	<i>Scenario 2 transformation for Rent-a-Car</i>	84
1.26.3	<i>Scenario 3 transformation for Rent-a-Car</i>	85
1.26.4	<i>Scenario 4 transformation for Rent-a-Car</i>	86
Chapter 7	DMT evaluation	87
1.27	<i>Meta-model compliance</i>	87
1.28	<i>SUMI survey and results</i>	90
1.29	<i>Individual participants' feedback</i>	92
1.30	<i>Improvements from feedback</i>	92
Chapter 8	Conclusion and future work	95
1.31	<i>Summary per research question</i>	95
1.32	<i>Future work</i>	98
1.33	<i>Concluding remarks</i>	98
Chapter 9	References	99

Chapter 10 Appendices 103
Appendix A 103
Appendix B 111
Appendix C 112

Chapter 1

Introduction

Enterprises in the modern age are continually growing in size and complexity. According to Boulding's (1956) nine-level complexity scale, the enterprise, as a social organisation, ranks eighth on the list, with only transcendental systems above it.

This complexity causes challenges within the enterprise. Oftentimes the modern enterprise develops in an ad hoc manner, which can result in misalignment within the enterprise as different sections and departments develop independently (de Vries, van der Merwe, & Gerber, 2017). This danger of misalignment leads to the conclusion that coherent thought and planning needs to go into the design of the enterprise. Thus enterprise engineering was born and defined as “the body of knowledge, principles and practices necessary to design an enterprise” (Giachetti, 2010).

Since the dawn of enterprise engineering, design approaches have been developed that aim to understand, describe and prescribe the design of an enterprise. Some prominent approaches include the Zachman approach (Zachman, 2003), the Open Group approach (Holt & Perry, 2010), and the Design & Engineering Methodology for Organisations (DEMO) approach (Dietz & Mulder, 2020). In this dissertation, the DEMO approach is examined.

Perinforma (2017) describes how the DEMO approach, developed by Professor Jan Dietz, works within the enterprise. Perinforma states that, in order to understand the enterprise as a whole, one must first understand the essence of the enterprise. She argues further that, at their core, all enterprises are the same. Thus, to engineer or design an enterprise, one must start with the essence. Dietz and Mulder (2020) describe the essence of the enterprise as revolving around the people in the organisation and how they interact with each other. These interactions are modelled using the four ontological models shown below in Figure 1.

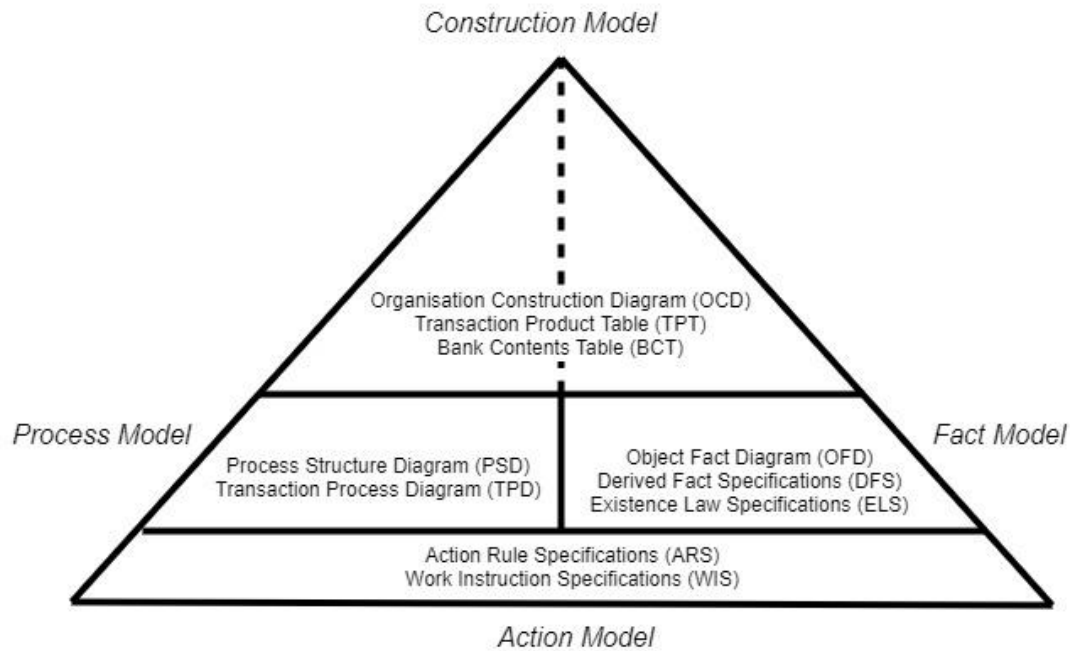


Figure 1: Ontological models (based on (Dietz & Mulder, 2017))

The four ontological models represented in Figure 1 are: The construction model (CM); the process model (PM); the fact model (FM), and the action model (AM). They are arranged in a triangular fashion to indicate the level of detail of each model. The CM is the most concise, and is used to specify the essence of the enterprise, whereas the other models are elaborations of the CM (Perinforma, 2017).

The CM, according to Decosse, Molnar, and Proper (2014), is useful for assigning responsibilities to users and individuals. The AM and FM, on the other hand, are used to develop or select applications. Of these three, only the CM is considered for the purposes of this dissertation.

The CM, in turn, is subdivided into three representations: the organisational construction diagram (OCD), the transaction product table (TPT), and the bank contents table (BCT). Only the OCD and TPT representations will be included in this dissertation. The BCT is excluded as it contains concepts used in the FM.

To model the processes within an enterprise more specifically, the Business Process Model and Notation (BPMN), as developed by the Object Management Group, can be used (Object Management Group, 2020). BPMN is used instead of the PM, as it is an industry-accepted modelling language (Grigorova & Mironov, 2014) that uses the same logic as the PM (Mráz, Náplava, Pergl, & Skotnica, 2017). A further advantage of BPMN is that it facilitates simulation

and workflow automation, as demonstrated in some BPMN-based tools such as ADONIS, Camunda, and Bizagi.

The goal of BPMN, as defined on the Object Management Group website (Object Management Group, 2020), is to provide a standard notation for business processes that is easily understood by business users while also containing complex process semantics for advanced users.

In this dissertation, the CM library for the OCD and TPT is created in the ADOxx platform. The existing BPMN 2.0 library is then used to facilitate the automatic transformation of some scenarios from the OCD to BPMN.

1.1 Problem context

This study was initiated due to a need at the University of Pretoria. Academics in the Industrial Engineering Department need to use the CM for both teaching and research purposes. The difficulty encountered was that there does not seem to be any modelling platform available that meets the university's requirements. Mulder (2019b) noticed a similar problem, and comments that there is limited automated support for DEMO modelling.

For a modelling platform to be useful for teaching, it needs to be *affordable* and *easy to use* so that students can learn the principles of enterprise modelling instead of struggling with an inefficient platform. Furthermore, given the present multi-view modelling paradigm in the literature, the CM needs to be vertically consistent with the other DEMO models. Consistency is important in multi-view modelling, as there are structural or semantic dependencies between different models of the same view (Bork, Buchmann, & Karagiannis, 2015). The multi-view modelling paradigm aims to make the representation of these systems easier to understand by representing a system from multiple viewpoints (Bork, 2016). It is thus important that the CM on the university's platform be vertically consistent with other DEMO models, ensuring that enterprise design is represented at different abstraction levels without having inconsistencies between these views.

Given the importance of being able to represent an organisation from different views, it would be useful to transform from the DEMO models to other types of models that are often used in conjunction with it. One such model that has already been mentioned is the Business Process Model and Notation (BPMN). BPMN is an industry standard (Grigorova & Mironov, 2014) and is used by several existing platforms, such as Bizagi, Camunda, and ADONIS, for simulation and business process automation. If a DEMO model could be horizontally transformed to a BPMN model without having to create BPMN models from scratch, it would save valuable time in the enterprise architecture and (re-)design process.

1.2 Problem statement

The problem is that no modelling platform is available that meets the University's requirements regarding the DEMO aspect models and model transformations. This limited automated support, as mentioned earlier by Mulder (2019b), is not just a problem at the University of Pretoria, but affects the wider enterprise engineering community as well.

Enterprise engineering is facing an increased emphasis on multi-perspective modelling (Bork, 2016). The multi-perspective modelling paradigm simply refers to the increasing importance of modelling from different perspectives, given the increasing complexity of entities to be modelled. The need for multi-perspective modelling can be seen in Section 1.9.3, where multiple authors have described how to transform from one model to another. There is, however, no modelling tool (software) available yet that facilitates the automatic or semi-automatic transformation of DEMO models to different models (see Section 1.9.4). A tool that enabled such transformation would greatly benefit the enterprise engineering community.

1.3 Research questions

For the purposes of this dissertation, the primary research question and thesis statement are defined according to the template prescribed by Wieringa (2014). The primary research question is then decomposed into secondary research questions. The secondary questions thus state the questions that are to be answered within the Master's study.

1.3.1 Primary research question

What DEMO tool, developed on the ADOxx platform, and based on a sound meta-model, will meet the university's requirements (including a *free-to-use* and *easy-to-use* tool for academics and students that *enables vertical consistency with other DEMO models*, and *enables horizontal transformation to other model types*, such as BPMN models), to help the University of Pretoria as well as the wider enterprise engineering community to address the emerging multi-view modelling paradigm?

1.3.2 Secondary research questions

Based on the primary research question, the following secondary questions are established.

Question 1: What are the available construction modelling tools? (Answered in Section 1.18)

Question 2: Do these tools meet the requirements of the University of Pretoria (the main stakeholder) – i.e., they are free-to-use, easy-to-use, enable vertical consistency with other DEMO models, and enable horizontal transformation to other model types? (Answered in Section 1.18)

Question 3: Does the problem instance feature as a class-of-problems in the existing literature? (Answered in Section 1.9.2)

Question 4: What are the key concepts used to describe modelling constructs and the translation or transformation of constructs? (Answered in Section 1.6)

Question 5: What are the vertical inconsistencies between different DEMO models, and how can they be avoided? (Answered in Sections 1.9.4 and 1.9.5)

Question 6: What approaches have been used to accommodate multi-view modelling up to now? (Answered in Section 1.9.3)

Question 7: Have other authors attempted to transform DEMO models into other model types, and if so, how was this done? (Answered in Sections 1.9.4 and 1.9.5)

Question 8: How should the new modelling tool be developed? (Answered in Chapter 5)

Question 9: How should the new modelling tool be evaluated? (Answered in Chapter 7)

Question 10: What are the constructional components of the tool? (Answered in Chapter 5)

Question 11: What case demonstrations provide sufficient evidence to demonstrate the utility of the tool? (Answered in Chapter 6)

Question 12: How useful is the tool in terms of its expected utility? (Answered in Chapter 7)

1.4 Dissertation statement

A new DEMO tool, developed on the ADOxx platform, and based on a sound meta-model that meets the requirements (including a *free-to-use* and *easy-to-use* tool for academics and students that *enables vertical consistency between DEMO models*, and that *enables horizontal transformation to other model types*, such as BPMN models) will help the University of Pretoria as well as the wider enterprise engineering community to model enterprises within a multi-view modelling paradigm.

1.5 Scope demarcation, limitations, assumptions

This study involves creating a tool in the ADOxx software platform that is capable of modelling the OCD and TPT representations of the CM, specified in DEMOSL 3.7, with the extensions described in Section 1.9.6. The initial version of the developed tool does not include any of the other DEMO aspect models. The new DEMO modelling tool (DMT) is also able to perform some horizontal transformations from the OCD to BPMN collaboration diagrams. Transformations are limited, based on the transformation specifications that were used. The existing BPMN 2.0 library in ADOxx was used, and not altered in any way.

It is to be noted that DEMOSL 3.7 was the newest version of the DEMO specification language when this tool was built. Given that the DEMO tool created in this dissertation forms part of the OMiLAB ecosystem, future developments by the community to accommodate the newer versions of DEMOSL, such as DEMOSL 4.5, are envisioned. More detail is also given on DEMO as a whole in the book by Dietz and Mulder (2020).

1.6 Definition of terms

The definitions and terms used in this dissertation, as identified from the literature in Chapter 2, are shown in Table 1 below.

Table 1: Definition of terms

<i>Term</i>	<i>Definition</i>	<i>Source</i>
Abstraction level	<i>The level of detail at which a model is described – i.e., a higher abstraction level can describe multiple models</i>	<i>(Bork, 2016)</i>
Aspect models	<i>Refers to the CM, PM, FM, and AM.</i>	<i>(Dietz & Mulder, 2017)</i>
BPMN	<i>Business Process Model and Notation</i>	<i>(Object Management Group, 2020)</i>
Collaboration diagram	<i>BPMN diagram modelling the collaboration between two or more participants (represented as pools)</i>	<i>(Von Rosing, von Scheel, & Scheer, 2014)</i>
DEMO	<i>Design & Engineering Methodology for Organisations</i>	<i>(Perinforma, 2017)</i>
DEMOSL	<i>DEMO specification language</i>	<i>(Dietz & Mulder, 2017)</i>
Horizontal consistency	<i>The consistency between models within the same development phase representing different views</i>	<i>(Bork, 2016)</i>
Meta-model	<i>Used to define sets of valid models and facilitate their transformation, serialisation, and exchange</i>	<i>(Mulder, 2019a)</i>
Modelling language	<i>The notation used to depict the graphical representation for each element; the syntax defining the grammar to be used; and the semantics specifying the meaning of the elements</i>	<i>(Bork, 2016)</i>
Multi-perspective OR Multi-view modelling	<i>Different models representing different views of the same system</i>	<i>(Awadid & Nurcan, 2019; Bork, 2016)</i>
Ontological models	<i>Refers to the CM, PM, FM, and AM</i>	<i>(Dietz & Mulder, 2017)</i>
Original production acts	<i>Refers to acts that create new or original facts</i>	<i>(Dietz & Mulder, 2020)</i>
Semantic consistency	<i>The behaviour of models should be semantically compatible with each other</i>	<i>(Awadid & Nurcan, 2019)</i>
Specification language	<i>Language used to specify the meta-model, rules, and legend of a modelling notation</i>	<i>(Mulder, 2019b)</i>
Syntactic consistency	<i>The models should conform to the abstract syntax specified by the meta-model</i>	<i>(Awadid & Nurcan, 2019)</i>
Transformation	<i>The process of generating a new view from a starting model (e.g., a DEMO organisation construction diagram to a BPMN collaboration diagram)</i>	<i>(Bork, 2016)</i>
Transaction pattern	<i>Refers to the general pattern occurring in human coordination. It is independent of the product that the coordination is about.</i>	<i>(Dietz & Mulder, 2020)</i>
Vertical consistency	<i>Consistency between models at different abstraction levels</i>	<i>(Bork, 2016)</i>

1.7 *Significance*

The problem mentioned in Section 1.2 is significant, as not solving it, impairs the University of Pretoria's development in teaching and research. As demonstrated in Chapter 4, none of the existing tools that were evaluated by the researcher meet the requirements set by the University of Pretoria (see Section 1.18). The problem analysis done in Chapter 4 indicates that, from the 12 tools investigated, none could model all four of DEMO's aspect models. Furthermore, only the tool developed in this dissertation is based on a coherent meta-model and facilitates some horizontal transformations from the DEMO construction diagram.

Solving this problem is also significant to the wider enterprise engineering community, especially as increasing emphasis is put on the multi-perspective modelling paradigm. In the systematic literature review (SLR) by Cicchetti, Ciccozzi, and Pierantonio (2019b), which is discussed in greater detail later (see Section 1.9.3), the importance of multi-view modelling and transformations within software is discussed. Therefore a tool that promises to facilitate automatic or semi-automatic transformations between DEMO and BPMN will benefit the modelling community in future.

Finally, in Section 1.19 it is shown that DEMO is widely taught in universities and educational institutions around the world. Therefore a tool that can model all aspect models (future vision) and perform transformations from DEMO to BPMN, which is an industry standard, promises to make a significant impact in both educational institutions and industry.

1.8 *Document structure*

The document is structured as follows. Chapter 2 investigates the relevant literature related to the research questions. It also highlights the key literature pertaining to the DEMO methodology and BPMN models. Chapter 3 describes the research methodology used in the dissertation. The expected rigour and ethical considerations are also discussed in this chapter.

Chapter 4 provides an in-depth analysis of both the problem faced and the solution requirements. Chapter 5 then continues by describing the design and development processes for DMT. This process is split into the facilitation of the CM within the ADOxx platform and the transformation process from the OCD diagram to the BPMN collaboration diagram.

In Chapter 6, two distinct cases are demonstrated within DMT. This is done first by creating the OCD and TPT diagrams for the case studies, and then transforming these into BPMN collaboration diagrams. Chapter 7 describes how both the usability and the technical adherence to specifications were evaluated. Finally, in Chapter 8, the conclusion of the dissertation as well as future research and development are discussed.

Chapter 2

Literature review

The purpose of this chapter is two-fold: (1) A systematic literature review (SLR) is used to gain insight into an existing class-of-problems and possible solution areas (see Section 1.9); and (2) recent literature on DEMO, the four aspect models, and BPMN is presented (see Sections 1.10 and 1.11). Finally, the literature on DEMO-to-BPMN transformations is investigated (see Section 2.4) with Sections 2.1, 2.2 and 2.3 providing the necessary context.

The SLR studies the existing literature in a systematic, complete, and reproducible way (Fink, 2005), and is used in this study to evaluate whether or not the problem features as a class-of-problems in the literature. It also serves as a way to concatenate available knowledge to aid in the development of a solution. The fact that the SLR is reproducible by other researchers adds to the reusability and credibility of the research done.

1.9 Class-of-problem validation and solution areas

In order to validate the class-of-problems and solution areas, *Research Questions 3, 4, 5 and 6* are answered by first doing an initial search for systematic literature reviews. The criteria used for these focused on issues of vertical and horizontal consistency between models, as well as horizontal transformation between models.

The following research review was found: “Consistency requirements in business process modelling: A thorough overview” by Awadid and Nurcan (2019). This literature study indicates a class-of-problems within the existing body of knowledge (*Research Question 3*).

Another research review was found: “Multi-view approaches for software and system modelling: A systematic literature review” by Cicchetti et al. (2019b). This review summarises approaches that accommodate multi-view modelling features in the available literature (*Research Question 6*).

To answer *Research Questions 4, 5 and 7*, a systematic literature review was completed to assess the consistency between the different DEMO aspect models, and whether other authors have attempted to transform DEMO models into other model types.

1.9.1 Research method for the systematic literature review

The research method followed for the class-of-problem validation and solution areas involved, extrapolates the relevant information from the existing literature review by Awadid and Nurcan (2019). Then a literature review was done concerning vertical inconsistencies between DEMO models, as well as the transformation of DEMO models into other model types.

The literature review by Awadid and Nurcan (2019) validates an existing class-of-problems – i.e., that both horizontal and vertical inconsistencies exist between business models – a general problem within the existing body of knowledge (*Research Question 3*). The initial keyword string used to search the literature was:

kw:(slr OR "systematic literature review" OR "literature review" OR overview) AND kw:("vertical consisten*" OR "horizontal* consisten*" OR consisten*) AND kw:("process model*" OR "business model*" OR "enterprise model*" OR "architecture model*" OR "business process model*") AND (eu:Peerreviewed)*

This was searched on the WorldCat database for the years 2018 to 2020. This produced 521 results. To narrow the search down, all the *keyword* operators in the search string were changed to *title*. This narrowed the result down to the single article of Awadid and Nurcan (2019). For the purposes of this dissertation, it is assumed that, since the review was published in 2019, the research is up to date.

The SLR done by Cicchetti et al. (2019b), on the other hand, discusses what is available in the existing literature regarding approaches for multi-view modelling (*Research Question 6*). This includes the area of existing modelling tools and how they accommodate both model creation and transformation. How consistency is maintained across the different views is also discussed. The initial keyword string used to find the SLR identified above pertaining to *Research Question 6* was:

kw:(slr OR "systematic literature review" OR "literature review" OR overview) AND kw:("multi-view" OR multiview OR "multi view") AND kw:(model) AND (eu:Peerreviewed)*

This was searched on the WorldCat database for the years 2018 to 2020. 424 results were produced. It was once again narrowed down by changing the *keyword* operators to *title*. This new search returned only the article by Cicchetti et al. (2019b), as shown above. The results obtained in the SLRs by Awadid and Nurcan (2019), and Cicchetti et al. (2019b) that are relevant to this dissertation are highlighted in Sections 1.9.2 and 1.9.3.

Then an *additional SLR* was done in this paper that aimed to answer the research questions specific to DEMO models, namely, *Research Question 4, 5 and 7*. The protocol for this literature study is described below. Knowing how previous authors attempted to answer these questions proved useful for finally solving the problem as a whole. The results for the class-of-problems and suggested solution areas of this review can be found in Sections 1.9.4 and 1.9.5 respectively.

The method followed for this new SLR was in accordance with the eight-step procedure described by Okoli and Schabram (2010). The criteria and techniques that were used for the

SLR are described in the protocol below, indicating how **steps 3-7** of Okoli and Schabram's study (2010) were performed.

Protocol for search (step 3): Here, it is detailed how the search for relevant literature is conducted. This protocol can be split into the following three sections: protocol for keywords, protocol for sources, and protocol for snowballing. The protocol for keywords highlights what strings of keywords are to be used in the search, while the protocol for sources indicates which sources will be used in the search.

Before searches were made regarding *Research Questions 5* and *7*, *Research Question 4* needed to be answered; that is, key concepts were defined from preliminary research into modelling constructs as well as the translation and transformation of constructs. These are listed in Section 1.6. The list of key concepts was then expanded as new terms were extracted from the examined literature.

With the important concepts and definitions defined above, searches were done to answer *Research Questions 5* and *7*.

The keyword string for *Research Question 5* is seen below:

kw:(consist OR inconsist*) AND kw:("DEMO method" OR "DEMO techni*" OR "DEMO methodology")*

The keyword string for *Research Question 7* is shown below:

kw:(conver OR transform* OR translat* OR change*) AND kw:("DEMO models" OR "DEMO method*")*

These strings were searched on the WorldCat and the SpringerLink databases. Both searches were initially conducted with the following added: (*AND ti:("SLR" OR "literature review")*), which yielded no results.

Within these searches, forward and backward snowballing were employed. Forward snowballing is used to search for papers that have cited the study in question, and backward snowballing to search the reference list of the selected studies. Snowballing was repeated until no more relevant studies were found.

Protocol for practical screen (step 4): The practical screen indicates what inclusion and exclusion criteria are used to vet which studies are to be considered.

The inclusion criteria were the following:

- *Only studies available at the libraries of the University of Pretoria and the University of Vienna were considered.*

- *Only studies from the year 2000 and onwards were considered.*
- *The search was performed in November 2019, and thus only studies up to that point were included.*
- *Only studies written in English were considered.*
- *Book sections, PhDs, Master’s theses, and conference papers were included.*

The exclusion criteria were these:

Studies whose abstract or title indicated that the content was not relevant to the research question were excluded. For example, ‘DEMO’ was used in the study as an acronym in a completely different field of interest.

Protocol for quality appraisal (step 5): The protocol for quality appraisal was used to determine which studies, after being practically screened, met the required quality to be included.

The quality appraisal consisted of the researcher reading through each study, and only including those that were relevant to the research questions. These studies needed to address the research questions either in part or in full.

Protocol for data extraction (step 6): It was imperative that the correct data was extracted from the sources selected. This was done by considering the research questions. Whenever a study mentioned the source of a specific inconsistency within the DEMO models or offered a potential solution, it was immediately recorded by the researcher (*Research Question 5*). Similarly, whenever a study mentioned a problem or potential technique regarding the transformation of a DEMO model into another type of model, the researcher recorded it (*Research Question 6*).

For relevant studies, *study title* and the *author* data were extracted. The researcher also recorded the *research question(s)* related to the identified study – i.e., whether the source aided in *validating the problem* or *offering a solution*.

Table 2 below shows the format in which the information captured in step 6 was stored.

Table 2: Example of extracting data for Step 6

<i>Author</i>	<i>Title</i>	<i>Code</i>	<i>Relation to theme</i>
Example person	Example Article 1	-	-

In this table the author and title of every study was shown. The ‘Code’ column refers to the codes defined in 1.9.4 and 1.9.5, where each code refers to a theme pertaining to either a class-of-problems or a solution area.

Protocol for data synthesis (step 7): The protocol for this step was split into two parts: (1) the techniques for the synthesis of studies to *validate a class of problems* (step 7a); and (2) the techniques for the synthesis of studies to *identify existing solution areas* (step 7b). This synthesis is done in Section 1.9.6, where the examined literature, along with its implications, is discussed.

1.9.2 Previous SLR results on class-of-problems

Question 3: Does the problem instance feature as a class-of-problems in the existing literature?

As discussed above, the SLR by Awadid and Nurcan (2019) indicates that there are both horizontal and vertical inconsistencies in the field of conceptual modelling. The relevant results from this SLR are summarised in Figure 2 below.

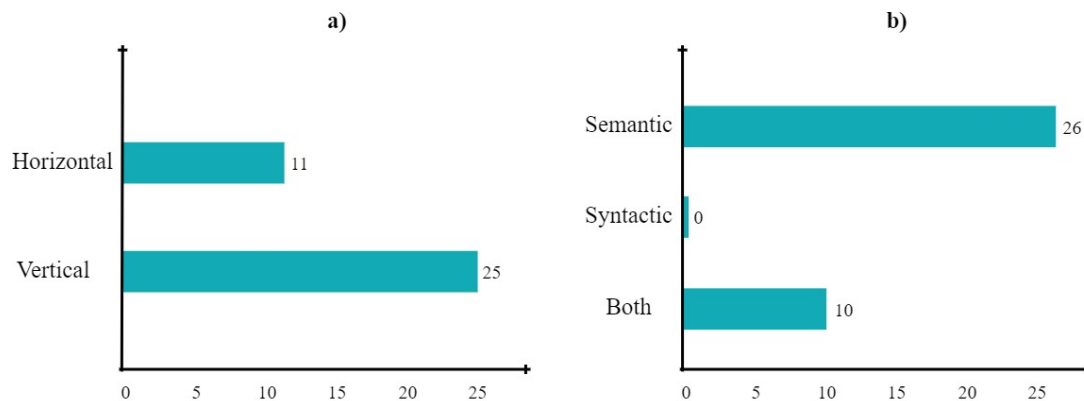


Figure 2: a) Type of inconsistency b) Nature of inconsistency

As can be seen in Figure 2, 25 of the literature sources identified by Awadid and Nurcan deal with horizontal inconsistencies, while only 11 deal with vertical inconsistencies. The right-hand side of Figure 2 indicates that 26 of Awadid and Nurcan's sources refer to semantics as being the nature of the inconsistency identified, while none of the inconsistencies are only syntactic in nature. Finally, 10 sources indicate that the inconsistencies are both semantic and syntactic in nature.

Awadid and Nurcan's (2019) SLR further shows that, in every instance where the nature of the inconsistency is both semantic and syntactic, the inconsistency type is horizontal. In fact, only one source is identified where only semantic inconsistencies are associated with horizontal inconsistencies.

Furthermore, all identified instances of horizontal inconsistencies take place when the context of the literature source studied is multi-perspective modelling. Vertical inconsistencies, on the other hand, occur either between variants of the same model at different abstraction levels, or when those variants of the model are merged.

It can thus be seen that both horizontal and vertical inconsistencies are prevalent as a class-of-problems in the existing literature.

1.9.3 Previous SLR results on multi-view modelling solutions

Question 6: What approaches have been used to accommodate multi-view modelling up to now?

The next SLR, as done by Cicchetti et al. (2019b), follows a slightly different approach to identifying appropriate studies. The authors of this SLR elected to select 10 core works in the literature in the field of multi-view modelling, and then to perform both forward and backward snowballing on these studies. Snowballing expanded the pool of primary studies to 40. The authors also decided to look only at papers that focus on some aspects of multi-view modelling and provide evidence of assessment of the approach. Furthermore, approaches that do not provide any conceptual, formal or technical solution for the proposed approach were excluded from the study. One of the studies that met these requirements , i.e. Bork & Karagiannis (2014) incorporates the ADOxx development platform used to develop DMT.

The first result of interest from the literature review is shown below in Figure 3. This figure indicates whether an approach supports the creation (i.e., the transformation) of multi-view artefacts. In the case of Figure 3, a value of “Yes” indicates that an approach does support multi-view artefact creation, while a value of “No” means it does not. The number of approaches that support multi-view artefact creation is only seven out of the 40. Six out of these seven approaches do, however, provide semi-automated support for the creation of multi-view models. An example of this would be a wizard in the modelling area leading the user to initiate a model transformation.

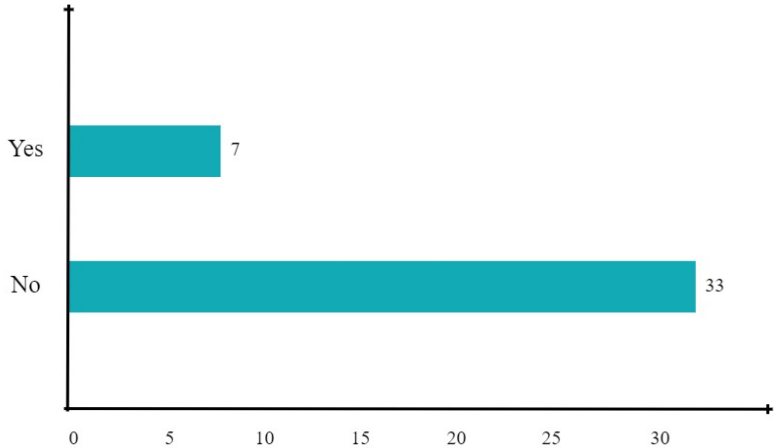


Figure 3: Support of multi-view artefact creation

Two essential attributes that the SLR investigates is whether and how previous authors have approached inter- and intra-model consistency. The authors of the review have found that 37 out of the 40 approaches do provide support for consistency management. Figure 4 below shows **a)** the number of approaches that provide support for intra-artefact consistency, and **b)** how many provide for inter-artefact consistency. Once again, the value “Yes” indicates that the approach does provide support for the relevant consistency.

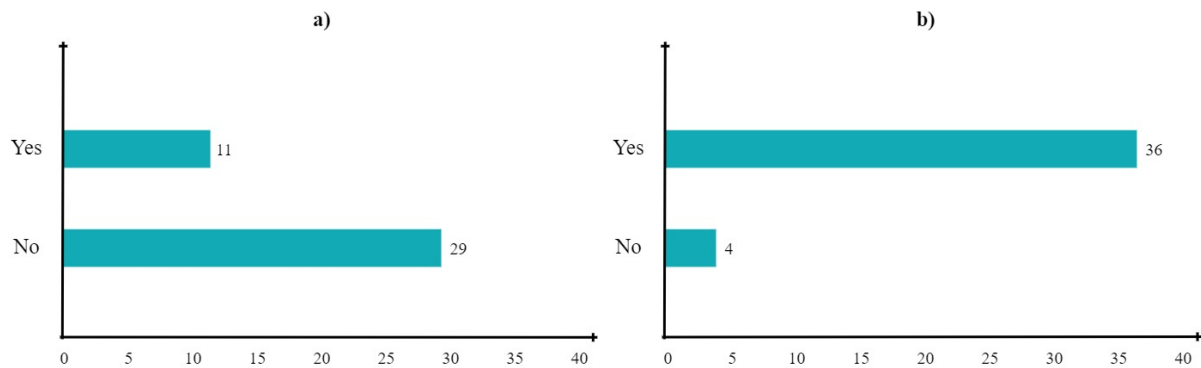


Figure 4: a) Support for intra-artefact consistency b) Support for inter-artefact consistency

As gathered from Figure 4 a), the majority of approaches do not provide support for intra-artefact consistency. For inter-artefact consistency, on the other hand, only four of the approaches do not provide support for consistency. The SLR further indicates that the inconsistency detection for inter-related artefacts is automated in 31 of the sources. It is worth noting that eight of the sources did not provide any information on inconsistency detection, while only one indicated that the inconsistency detection is done manually.

The SLR also identified the limitations of the existing approaches as they appear in the literature. The most pressing limitations, according to the SLR, are tool support (11 sources), consistency management (10 sources), and limited expressiveness to describe multi-view artefacts and the way they react (eight sources).

From the above section it can be seen that, so far, few multi-view modelling approaches have incorporated model transformations. The section also shows that, although the majority of approaches do take steps to address inconsistencies in multi-view modelling, it is still an area of concern for modelling tools.

From the SLR, the importance of tool support is most often highlighted in the literature. This does indicate that the modelling community expects not only a well-performing tool, but also background support.

Finally, the SLR found that a common problem in the approaches investigated is the lack of expressiveness in describing multi-view artefacts and the way they react. There is a need for more general functionality of modelling tools aside from just creating base models.

1.9.4 SLR results on DEMO class-of-problems

The search strings described above in Section 1.9.1 yielded the following results per research question.

Question 5: What are the vertical inconsistencies between different DEMO models, and how can they be avoided?

- *Initial search yielded 33 results*
- *Practical screening reduced this to 4 studies*
- *Quality appraisal reduced this to 2 studies*

Question 7: Have other authors attempted to transform DEMO models into other model types, and if so, how was this done?

- *Initial search yielded 83 results*
- *Practical screening reduced this to 6 studies*
- *Quality appraisal reduced this to 3 studies*

Added to the literature considered were the following articles. The first was acquired by contacting Mr Mulder about recent literature evaluating DEMO tools; the second article refers to the research-in-progress of which the author of this dissertation was aware.

1. *Validating the DEMO specification language* (Mulder, 2019b).
2. *Transforming organisation design knowledge into executable business processes: An approach based on DEMO and BPMN* (De Vries & Bork, In review).

Finally, one iteration of forward and backward snowballing was conducted using the above-mentioned literature and adhering to the same acceptance criteria described above. This increased the number of literature documents to 23. These are shown in Table 3 below.

Table 3: Literature documents

<i>Authors</i>	<i>Title</i>
<i>Albani and Dietz (2006)</i>	The benefit of enterprise ontology in identifying business components
<i>Badal (2009)</i>	A roadmap for the transition from paper to digital records
<i>Bakhshandeh (2016)</i>	Ontology-driven analysis of enterprise architecture models
<i>Boedhrum and Algoe (2009)</i>	DEMO applied to financial services
<i>de Kinderen, Gaaloul, and Proper (2014)</i>	Bridging value modelling to ArchiMate via transaction modelling
<i>De Vries and Bork (In review)</i>	Transforming organisation design knowledge into executable business processes: An approach based on DEMO and BPMN
<i>Ding (2016)</i>	Integrating value modeling into ArchiMate
<i>Ettema and Dietz (2009)</i>	ArchiMate and DEMO – mates to date?
<i>Figueira and Aveiro (2014)</i>	A new action rule syntax for DEMO models based automatic workflow process generation (DEMOBAKER)
<i>Geskus (2008)</i>	DEMO applied to quality management systems
<i>Granjo, Bakhshandeh, Pombinho, da Silva, and Caetano (2014)</i>	Validating value network business models by ontologies
<i>Kharmoum, El Bouchti, Ziti, and Omary (2019)</i>	Descriptive analysis of business value models' transformation in MDA approach
<i>Lara, Sánchez, and Villalobos (2019)</i>	OT modelling: the enterprise beyond it
<i>Mráz et al. (2017)</i>	Converting DEMO PSI transaction pattern into BPMN: A complete method
<i>Mulder (2019b)</i>	Validating the DEMO specification language
<i>Pombinho, Aveiro, and Tribolet (2014)</i>	A matching ontology for e3Value and DEMO – A sound bridging of business modelling and enterprise engineering
<i>Seck and Barjis (2015)</i>	An agent based approach for simulating DEMO enterprise models
<i>Singh (2013)</i>	Integrating business value in enterprise architecture modelling and analysis
<i>Van Kervel (2012)</i>	Ontology driven enterprise information systems engineering
<i>Vejraskova and Meshkat (2013)</i>	Translating DEMO models into Petri Net
<i>Wang (2009)</i>	Transformation of DEMO models into exchangeable format
<i>Wang, Albani, and Barjis (2011)</i>	Transformation of DEMO metamodel into XML schema
<i>Yamamoto, Olayan, and Morisaki (2019)</i>	Analyzing e-health business models using actor relationship matrix

To identify the class-of-problems indicated in the literature, the code scheme below is made, showing the codes as extrapolated from the literature indicated in Table 4. As can be seen in the table, both the code and the code names are shown for *Research Questions 5* and *7*. In Table 4 the following abbreviations are used to define the codes:

CCP: Consistency class-of-problem

TCP: Transformation class-of-problem

Table 4: Coding scheme: Class-of-problem

<i>Research Question 5</i>		<i>Research Question 7</i>	
<i>Code:</i>	<i>Code name:</i>	<i>Code:</i>	<i>Code name:</i>
CCP 1	Meta-model inconsistency	TCP 1	Software support
		TCP 2	Transformation limitation

Given the above table, a codebook is generated for the codes described. This codebook is based on the example provided by Guest, MacQueen, and Namey (2011) and shows the code name, gives a description of the code, and indicates when and when not to use the code. Both this codebook and the codebook for solution areas were verified for clarity by having an independent coder code two of the articles considered. The intercoder agreement is shown in Appendix B. The codebook can be seen directly below.

Code CCP 1

Code name: Meta-model inconsistency

Brief definition: Inconsistency due to meta-model

Full definition: There is inconsistency within DEMO due to the meta-model being inconsistent.

When to use: Whenever it is suggested or stated that DEMO's meta-model or specification language (DEMOSL) is inconsistent. It could also state that the meta-model is incomplete, inadequate, needs improvement, is incapable, cannot capture, cannot model, or anything similar.

When not to use: When a change is proposed to the current meta-model (See CSA3: DEMOSL external consistency) or a new meta-model (See CSA4: External consistency) is proposed.

Code TCP 1

Code name: Software support

Brief definition: Limitation in software support

Full definition: There is limited or no software support for performing the transformation of DEMO models to other models.

When to use: When a tool does not cater for the semi-automatic or automatic transformation of DEMO models to model-views of a different modelling language. This can also be used when a transformation technique is described, but no reference is made to software supporting this transformation.

When not to use: When discussing a limitation or difficulty in the transformation technique itself (See TCP2), or when the tool platform is scrutinised due to factors other than the transformation.

Code TCP2

Code name: Transformation limitation

Brief definition: Limitations in transformation

Full definition: There is some difficulty or limitation in the transformation of the DEMO model.

When to use: When there is a theoretical limitation or difficulty in transforming the DEMO model to a different model type. This could be due to the transformation being incomplete, inaccurate, or simply limited in the types of situation it can address.

When not to use: When discussing the imperfections of the actual modelling languages, as opposed to the difficulty in the transformation. Also, when a software tool is limited in its capacity to transform the DEMO model, as opposed to a limitation in the actual transformation specification.

These codes are identified as general themes found in the literature pertaining to a class-of-problems. The results of the SLR using these codes can be seen in Table 5 below. This table presents and briefly discusses the extracted themes of *Research Questions 5 and 6* from the literature in the format prescribed in Table 4 above.

Table 5: Results table: Class-of-problem

<i>Research Question 5</i>		
<i>Number of sources:</i>	<i>Code:</i>	<i>Example of quote:</i>
1	CCP 1	<i>“_analysis of the metamodel showed some inconsistencies.” (Mulder, 2019b)</i>
<i>Research Question 7</i>		
16	TCP 1	<i>“Thus, we showcase the possibility for creating a formal tooling chain, whereby models created in a can be exported to a format interpretable for an ArchiMate tool.” (de Kinderen et al., 2014)</i>
1	TCP 2	<i>“DEMO transaction patterns introduce bias towards social perspective” (de Kinderen et al., 2014)</i>

1.9.5 SLR results on DEMO-related solution areas

The steps followed for determining the SLR results for the *solution areas* are the same as with the class-of-problems. First, the coding scheme is established, as can be seen in Table 6 below. This once again shows, for both research questions, the code and the code names used. The abbreviations used for the codes are as follows:

CSA: Consistency solution area

TSA: Transformation solution area

Table 6: Coding scheme: Solution area

<i>Research Question 5</i>		<i>Research Question 7</i>	
<i>Code:</i>	<i>Code name:</i>	<i>Code:</i>	<i>Code name:</i>
CSA 1	Design consistency	TSA 1	Concept transformation
CSA 2	Update consistency	TSA 2	Direct concept transformation
CSA 3	DEMOSL External consistency		
CSA 4	External consistency		

Once again, a codebook is generated according to the information on Table 6. This codebook is shown below.

Code: CSA 1

Code name: Design consistency

Brief definition: Consistent by design

Full definition: DEMO is already consistent by design, and has no inconsistencies between the DEMO models.

When to use: Use this for all references stating that DEMO is consistent by design. These references might refer to DEMO as consistent, or as meeting the C4E requirements; or they could be stated as a negative – for example, “DEMO is not inconsistent”.

When not to use: Do not use this code when DEMO is said to have become consistent due to an update in the DEMOSL language (see CSA 2: Update consistency) or another external improvement of some kind.

Code: CSA 2

Code name: Update consistency

Brief definition: Consistent due to update

Full definition: DEMO is consistent due to an update to the DEMOSL.

When to use: Use this whenever a reference states/infers that DEMO has become consistent due to an update to the DEMOSL. This could refer to general improvements between versions, general updates, the links between the models, or the notation of the language.

When not to use: Do not use this code when an update is proposed to the DEMO language, or when the language is altered or extended to accommodate an inconsistency (See CSA 3: DEMOSL external consistency).

Code: CSA 3

Code name: DEMOSL external consistency

Brief definition: Consistent due to external influence on the DEMOSL

Full definition: DEMO is consistent due to proposed external influence, either by altering the DEMOSL or by adding to it.

When to use: Whenever a change or an update is proposed to the DEMOSL in order to improve consistency.

When not to use: When the suggested solution does away with the DEMOSL altogether (See CSA 4: External consistency).

Code: CSA 4

Code name: External consistency

Brief definition: Consistent due to external influence

Full definition: DEMO is consistent due to proposed external influence that does away with the DEMOSL.

When to use: Whenever the proposed solution does away with DEMOSL altogether. This could be by proposing a new specification language or meta-model.

When not to use: When the suggested solution simply changes or updates DEMOSL (See CSA 3: DEMOSL External consistency).

Code: TSA 1

Code name: Concept transformation

Brief definition: Transformation from DEMO via concept identification

Full definition: The DEMO model is transformed to another model type by using the meta-models' conceptual relationships.

When to use: Whenever conceptual relationships are used to transform DEMO to another type of model via the meta-model. This refers to the logic of identifying concepts from the two meta-models that are similar, and transforming from one model to the other accordingly. Different terms can be used for this, such as conceptual translating, conceptual transformation, or conceptual conversion.

When not to use: When the transformation happens from a different model to the DEMO model. Also, don't use when the meta-model is not used in the transformation (See TSA 2: Direct concept transformation).

Code: TSA 2

Code name: Direct concept transformation

Brief definition: Transformation from DEMO via model concepts

Full definition: The DEMO model is transformed to another model type by identifying similar concepts to link the models.

When to use: When the DEMO model is transformed to another model by identifying similar concepts in the models and linking them. This could be by identifying similar functions, actions or actors.

When not to use: When this conceptual linking is done via the meta-model (See TSA 1: Concept transformation) or when transformation is done from a different model to the DEMO model.

These codes are identified as general themes found in the literature pertaining to solution areas. The results of the SLR using these codes can be seen in Table 7 below. This table once again presents and briefly discusses the extracted themes from the literature. Given the number of documents considered, the table summarises the results by showing the number of sources that pertain to each theme. Appendix C provides greater detail on each source.

Table 7: Results table: Solution areas

<i>Research Question 5</i>		
<i>Number of sources:</i>	<i>Code:</i>	<i>Example of quote:</i>
8	CSA 1	<i>“The ontological model of an enterprise is guaranteed coherent, consistent, comprehensive, and concise.” (Geskus, 2008)</i>
1	CSA 2	<i>“This notion is not consistent with the FM, therefore, it has been changed in DEMOSL 3.7.” (Mulder, 2019b)</i>
1	CSA 3	<i>“expansion of DEMOSL ... by expanding the specification to all model aspects that need to be described.” (Mulder, 2019b)</i>
3	CSA 4	<i>“of DMOL models and for DEMO models.” (Van Kervel, 2012)</i>
<i>Research Question 7</i>		
15	TSA 1	<i>“We use only the concept mappings described above.” (de Kinderen et al., 2014)</i>
1	TSA 2	<i>“We have presented the DEMO methodology in some detail and translated ... into the agent-based modeling paradigm.” (Seck & Barjis, 2015)</i>

1.9.6 Discussion

The SLR establishes what is written in the literature regarding a class-of-problems and potential solution areas associated with the DEMO modelling language (addressing *Research Questions 5 and 7*). It does this by extrapolating themes from the literature with the use of codes established in the code book. These themes are discussed below for each research question.

The only class-of-problems identified as answering *Research Question 5* is that relating to the meta-model inconsistencies that exist in the DEMOSL 3.7. These inconsistencies are identified

by Mulder (2019b), and include inconsistencies within each aspect model as well as overall inconsistencies that effect the relationships between different aspect models.

Regarding solution areas, answering *Research Question 5*, eight of the identified sources refer to DEMO as being consistent or as claiming to be consistent in the way it models enterprises' organisation. While this might seem to contradict the class-of-problems, it does not. The inconsistency in the meta-model indicated above does not affect the consistency between DEMO models when they are modelled separately. It is sure, however, to cause inconsistencies where automatic or semi-automatic transformation between the models are concerned, as indicated by Mulder (2019b).

Both the solution areas pertaining to codes CSA 2 and CSA 3 are proposed by Mulder in his study. None of the other studies examined proposed solutions within these themes. Regarding CSA 2, Mulder states some inconsistencies that were present in DEMOSL 3.6 but that were rectified in DEMOSL 3.7. Given that Mulder is a co-author of DEMOSL 3.7, this suggests that the inconsistencies identified in DEMOSL 3.7 are likely to be fixed in the future. This leads to CSA 3, where Mulder suggests in the article that DEMOSL 3.7 be extended to cater for the additional views needed in the meta-model.

Finally, regarding CSA 4, there are two distinct cases where authors have constructed different meta-modelling languages for DEMO. Van Kervel (2012) claims that DMOL, his meta-model, satisfies the requirements for 4C-ness (coherent, consistent, comprehensive, and concise). Wang (2009), on the other hand, claims that his version of the meta-model provides a complete representation of all models.

Despite of these two possibilities for new meta-models, the literature review does not indicate that they have been used again or updated. Taking this into consideration, and that DEMOSL has been updated periodically to account for any inconsistencies in the language, DEMOSL 3.7, with the extensions proposed by (Mulder, 2019a) for the CM, was chosen as the meta-model to be used for DMT, as it was the newest and most consistent version available when development started.

It is worth noting that, since the initial completion of this literature review, a newer version of the DEMO specification language (DEMOSL 4.5) has been released (<https://demo.nl/mdocs-posts/2019-12-25-demo-specification-language-4-0/>). This has not been incorporated into the DMT tool created in this dissertation, but will be included in a future update to the tool.

For *Research Question 7*, the most prevalent class-of-problems identified in the literature pertains to software automation in transformation (TCP 1). Even though some sources, such as de Kinderen et al. (2014), indicate that transformation automation is potentially feasible by way

of modelling software, none of the sources attempt to automate the transformation of DEMO models to other models.

This is interesting, given that the literature review by Cicchetti et al. (2019b), discussed in Section 2.2.3, indicates that only six of the 40 sources studied allow for the automated or semi-automated creation of multi-view models. It can thus be seen that, although there is a need for the transformation in multi-view modelling, few known software tools exist that facilitate this, none of which pertain to DEMO modelling.

The only other potential class-of-problems, related to *Research Question 7*, is identified by de Kinderen et al. (2014). Here they refer to a potential pitfall of transforming from DEMO to ArchiMate – specifically, that DEMO could cause a bias towards the social perspective.

The vast majority of sources pertaining to solution areas make use of concept transformations (TSA 1). This refers to the transformation from DEMO to another model by matching concepts from DEMO's meta-model to the meta-model of the other model.

The specific transformations done in the literature using concept transformations are:

- *DEMO to ArchiMate* (de Kinderen et al., 2014): Here DEMO is used as a bridge between the e³value model and ArchiMate. The authors of the paper use conceptual mapping to perform this transformation.
- *DEMO to Petri Net* (Vejrazkova & Meshkat, 2013): In this paper the authors aim to convert DEMO models into Petri Net models in an attempt to gain insight into a model's behaviour.
- *DEMO to XML* (Wang, 2009): Here the author creates transformation rules to transform DEMO models to XML models in order to focus on platform-specific requirements. This is done by mapping DEMO concepts to corresponding XML concepts via the DEMO meta-model. Although this is done within a software application with clear instructions, it requires such a large amount of user input that the transformation cannot be seen as automatic or even semi-automatic.
- *DEMO to BPMN*: Four sets of authors are identified who investigate the transformation of DEMO to BPMN models. This is the specific transformation in which this dissertation is interested. Therefore these sources are discussed in greater length (see Section 1.12) once more background on the DEMO aspect models and underlying theories has been introduced.

Finally, direct concept transformation (TSA 2) is not considered as viable for this dissertation. The reasoning is as follows. For this solution area, Seck and Barjis (2015) transformed a specific instance of the DEMO model into a simulation. The difference between their approach and those described above is that they transformed the model by directly translating the terms in a specific instance of a DEMO model to an agent-based simulation. As indicated by Seck

and Barjis (2015), their approach transforms concepts to facilitate agent-based simulation, whereas our transformation approach, incorporating BPMN, applies a discrete event paradigm.

From the discussion above, it can be concluded that both *Research Questions 5* and *7* have been answered. DEMO can be concluded to give a consistent view when each aspect model is modelled individually; but inconsistencies in the meta-model could cause inconsistencies between the models when automatic or semi-automatic transformations are done. DEMOSL 3.7, with extensions, is thus chosen as the meta-model for this dissertation as the most recent specification at the time of development.

1.10 Theory on DEMO aspect models

Since the main objective is to develop a new DEMO modelling tool (called DMT) that facilitates model transformations, a thorough understanding is needed of the DEMO aspect models and some kernel theories. Since DMT was based on DEMOSL 3.7, supporting theory from Perinforma (2017) is presented. Additional insights presented by Dietz and Mulder (2020) have also been included to provide additional clarification on theories and concepts. In Sections 1.10.1 and 1.10.2 the PSI theory and the ALPHA theory are discussed. These theories provide some of the building blocks on which the DEMO methodology is built. The DEMO aspect models are then introduced in Section 1.10.3.

1.10.1 The PSI theory

The PSI (*Performance in Social Interaction*) theory is a study of the operational essence of an organisation. Given that organisations are social systems, this essence relates to people and how they interact with each other. People (subjects) are referred to as *actors* who have authority and responsibility to perform a *role*. In completing their roles, commitments between the actors manifest as *coordination acts* and subsequent *coordination facts* (Dietz & Mulder, 2020).

The PSI theory is relevant to this study, as it presents a consistent way of modelling the coordination between actors in an enterprise. The most basic representation of this is the *basic transaction pattern* demonstrated below in Figure 5.

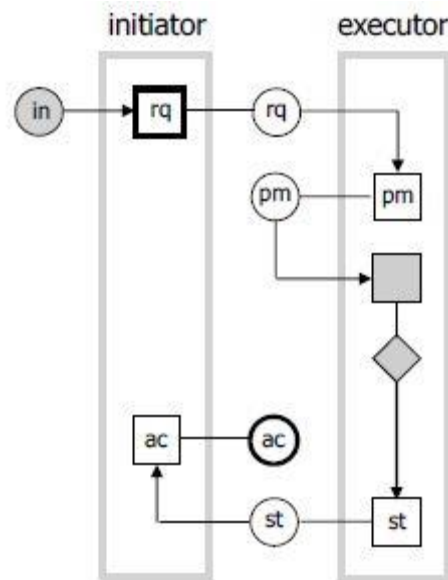


Figure 5: Basic transaction pattern (Perinforma, 2017)

The basic transaction pattern aims to model the coordination acts and facts created between actor roles. The two specified actor roles are the *initiator* and the *executor*. The transaction pattern thus starts in an initial (in) status, followed by the initiator performing the act of *requesting* (rq) the executor to perform an action. The executor then *promises* (pm) to perform the action, performs it, and then *states* (st) that s/he has completed it. Finally, the initiator *accepts* (ac) that the task has been completed as promised (Perinforma, 2017).

The limitation in the basic transaction pattern is that it only accounts for a positive response from both parties. The *standard transaction pattern*, represented by the central process in Figure 6, expands the basic transaction pattern by adding negative responses to the coordination acts. Thus, instead of promising to perform a task, the executor can *decline* (dc) the request. This leads the initiator, after further negotiation, either to restart the process with a new request, or to *quit* (qt) the transaction, effectively returning the process to its initial state.

The same logic holds true for the *state* and *accept* steps. The initiator can now *reject* (rj) the statement of the executor that the task was completed as promised. The executor can then either *restate* task completion after discussion with the initiator, or *stop* (sp) the process and return it to initial state.

The *complete transaction pattern*, shown in Figure 6, attempts to further accommodate human behaviour by modelling a person's tendency to change their mind. Thus the four corner processes in the figure illustrate the acts of revoking any steps in the transaction pattern. A practical example of this might be an executor promising to perform a task, but then realising that they do not have the equipment or skillset to do this. They then revoke the promise (rv pm)

to the initiator. After this step, negotiations can take place either to restart the transactional process or to return it to its initial state.

The complete transaction pattern is important, as the entire pattern is an intrinsic concept that is included in the DEMO's aspect models. For instance, the OCD diagram in the CM consists of actors who take on the role of executing or initiating a transaction. The OCD becomes a pattern that represents multiple instances of the complete transaction patterns that are executed during an enterprise operation.

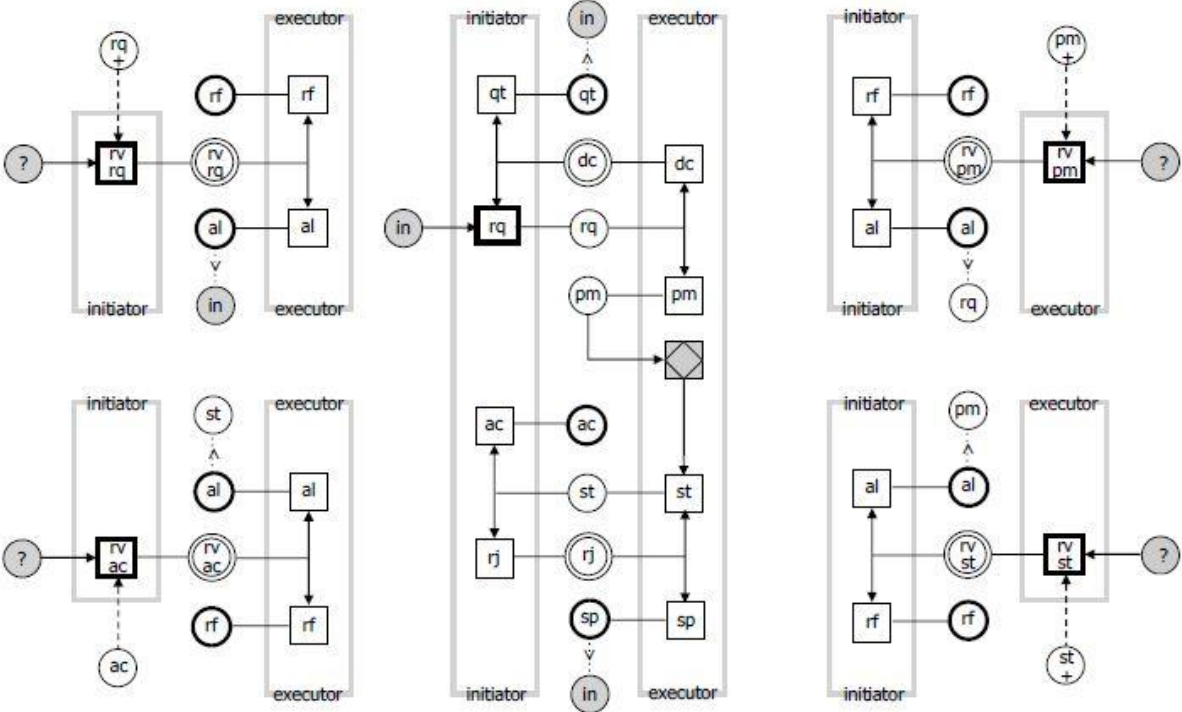


Figure 6: The complete transaction pattern (Perinforma, 2017)

1.10.2 The ALPHA theory

The ALPHA (*Abstraction Layers in Production for Holistic Analysis*) theory, as stated by Dietz and Mulder (2020), investigates the different levels of *production acts* that an *actor role* performs. As shown in Figure 7 below, there are three different levels of production acts. These are represented by the different sections of the tree displayed in Figure 7. They are, reading from top-to-bottom, *original production acts* (red); *informational production acts* (green), and *documental production acts* (blue).

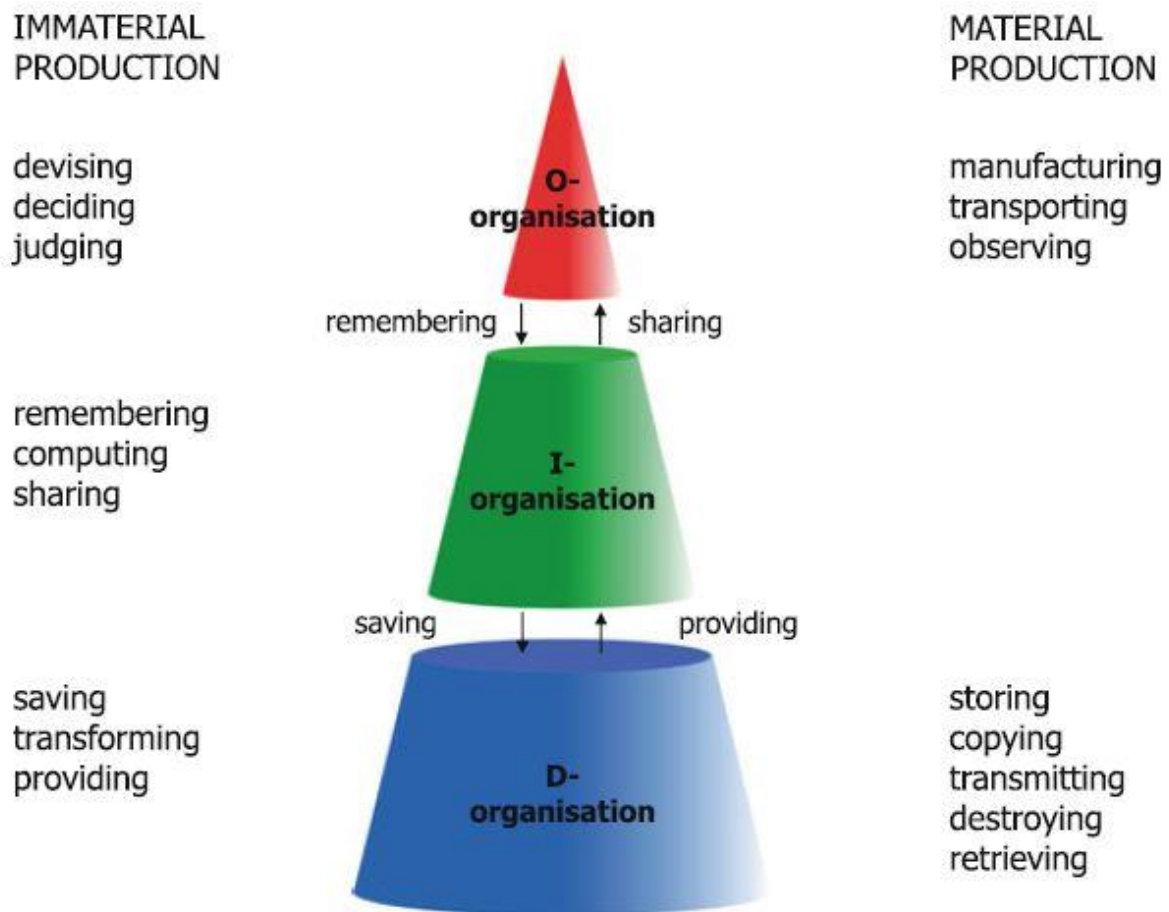


Figure 7: Organisation layers and types of production acts (Dietz & Mulder, 2020)

The *original production acts* are those that bring about new, or original, *production facts*. These could be immaterial acts, such as deciding or devising things, or they could be material acts, such as manufacturing or transporting something. Here it is important that the actor has performed acts that change the state of the production world, and thus needs to be a responsible and authorised actor.

Informational production acts consist of remembering, computing, or sharing production facts already created. Thus informational acts do not alter the state of the production world, but rather represent the information in a different format.

Documental production acts concern facts, and provide a means to process and transport these facts, such as storing, copying, and transmitting them. The documental acts can also transform or save the facts.

The DEMO CM assumes that only the *original production acts* will be modelled. The red, green and blue colour differentiation between different act types is not included in DMT, but future versions of the tool will also accommodate informational production acts and documental production acts.

1.10.3 The essential model of an enterprise

The four aspect models (CM, PM, AM, FM), already briefly introduced in Chapter 1, are hierarchical in nature, as shown in Figure 8. The most concise model, the CM, is located at the top of the triangle, as it has the highest level of abstraction and is thus the most concise. This model is discussed first.

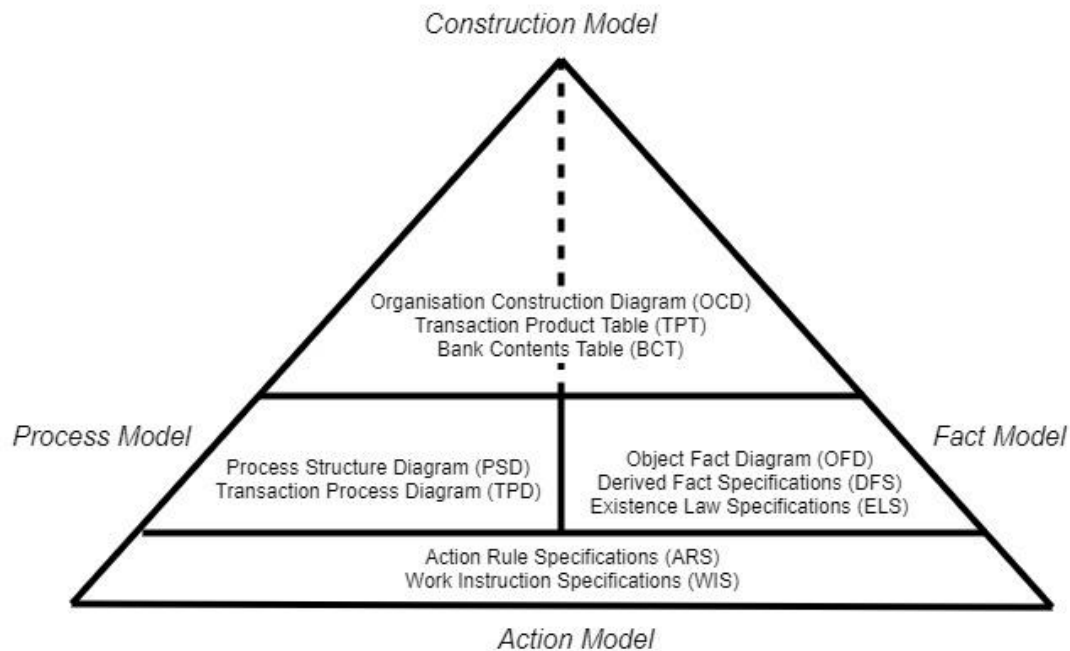


Figure 8: Ontological models (based on (Dietz & Mulder, 2017))

To aid the explanation of the aspect models, an example by Perinforma (2017) of a tennis membership at the club ‘Volley’ is used. This is a good all-encompassing example, as all of the aspect models – as well as how they are dependent on each other – are demonstrated.

The CM

The CM is sub-divided into three representations, OCD, TPT, and BCT, which represent the essence of the enterprise as the highest level. The OCD consists of actor roles and transaction kinds connected via *executor* and *initiating links*. Together these represent a consolidated version of the complete transaction pattern found in Section 1.10.1 (Perinforma, 2017). This is illustrated in Figure 9.

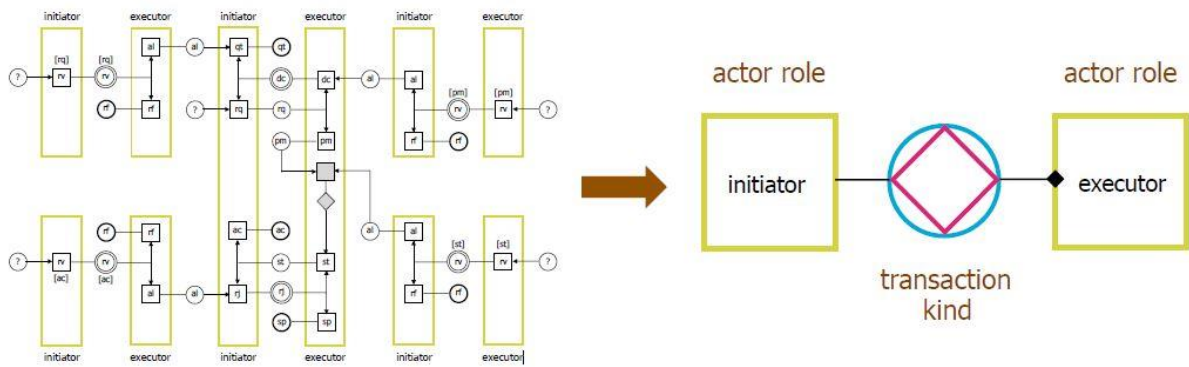


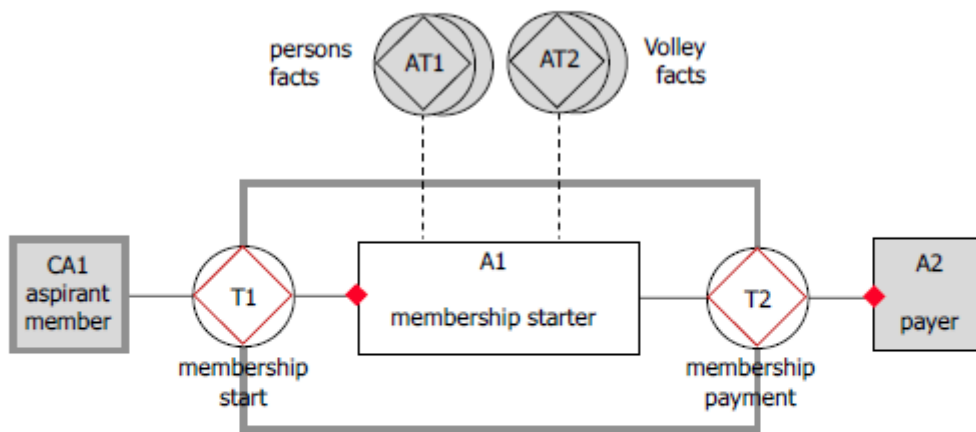
Figure 9: The complete transaction pattern in the CM (Perinforma, 2017)

An example of an OCD and a TPT is shown in Figure 10. Each *transaction kind* (TK) needs to be executed by one actor role, and can be initiated by multiple *actor roles* (ARs). An AR can also be the initiator of multiple TKs. In the case in Figure 10 below, the TKs are all only initiated by one AR. The TKs are modelled within the context of a *scope of interest* (SOI).

There may be sub-networks of ARs and TSs that the modeller wishes to group together. These can be modelled as *composite actor roles* (CARs), and can be represented as either white box or black box elements, depending on whether the modeller is interested in the details of the sub-network. CARs within the SOI are discussed further in Section 1.23.

TKs can be grouped together in a similar way to CARs, and are classified as *aggregated transaction kinds* (ATKs). In practice these are modelled outside the SOI, and can be connected to the SOI or any element inside the SOI by an *information link*.

Here the process of the already-mentioned ‘Volley’ is shown. In the OCD, a black-box CAR (*aspirant member*) can be seen, as well as the actor roles (*membership starter; payer*) and transaction kinds (*membership start; membership payment*). An SOI and two ATKs are also modelled, where both ATKs are connected to the ‘membership starter’ actor role. The ‘membership starter’ thus has access to, and uses information provided by, the ‘persons facts’ and ‘Volley facts’ (Perinforma, 2017).



transaction kind	product kind
T1 membership start	P1 Membership is started
T2 membership payment	P2 the first fee of Membership is paid

Figure 10: Example of OCD and TPT (Perinforma, 2017)

The TPT, as seen in Figure 10, shows the TKs present in the OCD, as well as their *product kinds* (PKs). The TK ID (e.g., T1 and T2) and the PK ID (e.g., P1 and P2) are also included in the TPT. The PK communicates the product created for an instance of a TK (Perinforma, 2017). Finally, the BCT is introduced, thus completing the CM. The BCT of ‘Volley’ is shown below in Figure 11. This illustrates that certain facts are produced by certain TKs. For instance, from the figure, some dependent and independent facts pertaining to ‘MEMBERSHIP’ is associated with T1, including facts such as the starting day, the member, and the amount due of a membership instance (Perinforma, 2017).

bank	independent/dependent facts
T1 membership start	MEMBERSHIP Membership is started the starting day of Membership the member of Membership the amount to pay of Membership
T2 membership payment	the first fee of Membership is paid the amount paid of Membership
AT1 persons facts	PERSON the day of birth of Person
AT2 Volley facts	[YEAR] the minimal age in Year the annual fee in Year the max members in Year

Figure 11: Bank contents table (Perinforma, 2017)

The PM

The PM is discussed next. It consists of a *process structure diagram* (PSD) and a *transaction process diagram* (TPD). The PM is placed between the CM and AM on the hierarchy in Figure 8, and is thus more detailed than the CM and less detailed than the AM. It serves to model the *state* and *process view* for the coordination world. Here, ‘state view’ refers to the state of the world, whereas ‘process view’ refers to the transitions between states.

The PSD for the volley example is shown in Figure 12 below. Here, the TKs are drawn in shapes akin to sausages, which are spread between the relevant actor roles (modelled as swim-lanes). Viewing the PSD from left to right, the sequence of events is shown. The production acts and facts are represented on the figure by bordered boxes and disks respectively. The occurrence of the production acts is also shown on the diagram, represented by the diamond within each TK (Perinforma, 2017).

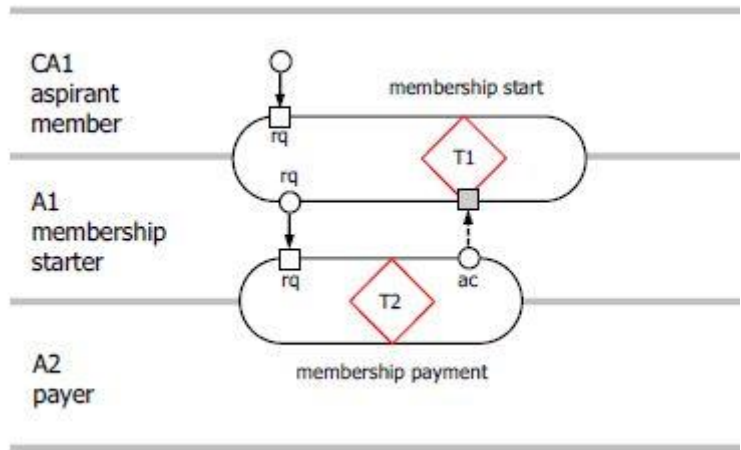


Figure 12: Process structure diagram (Perinforma, 2017)

Also included in the PM is the TPD. This, aside from showing the complete transaction pattern found within the TK, also shows the links between TKs. For example, the TPD for T1 in the continued example of ‘Volley’ is demonstrated in Figure 13 (Perinforma, 2017).

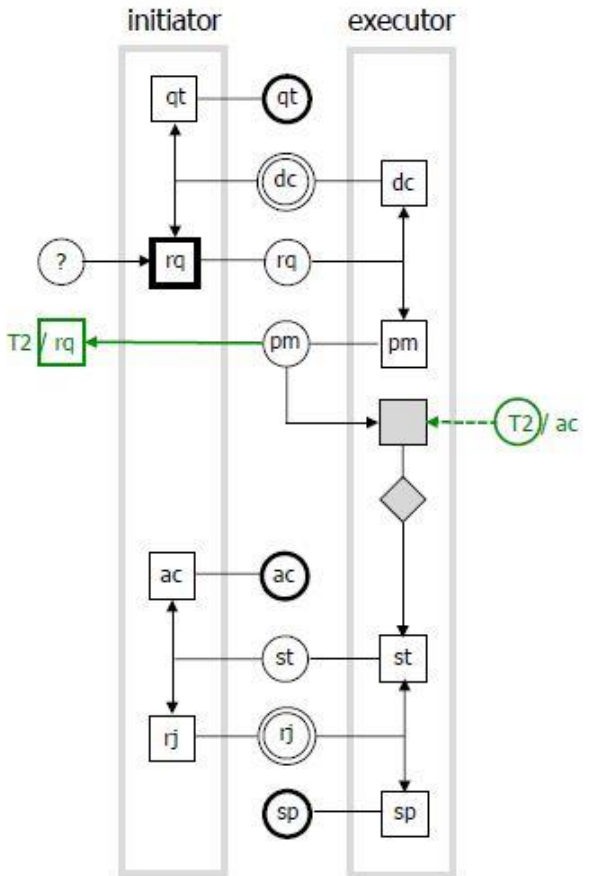


Figure 13: Transaction process diagram (Perinforma, 2017)

Not only is the complete transaction pattern between initiator and executor shown in Figure 13, but the links between T1 and T2 are also shown. T2 is initiated when the executor in T1

promised to perform the act of starting a membership. It is further shown that the executor in T1 must wait for the membership payment to be accepted before it can execute the task of starting the membership.

The PM can thus be seen to elaborate on the CM by specifying the sequence in which events occur, and exactly how TKs are dependent on each other (Perinforma, 2017).

The FM

The FM consists of the *object fact diagram* (OFD), *derived fact specifications* (DFS), and *existence law specifications* (ELS). As stated by Perinforma (2017), it aims to model the state and process views for the *production world*. The FM differs from the PM, since the PM models the state and process views for the *coordination world*. The CM and AM, in turn, incorporate both the coordination world and the production world.

The OFD for the volley membership example can be seen in Figure 14. Here, the round-cornered rectangles represent *object classes* (MEMBERSHIP; PERSON; PAID MEMBERSHIP). Object classes can be seen as an extension of a fact type. This extension can be noted by referring back to Figure 11 and observing that the fact types created in the BCT match the object classes in the OFD below. Why the entity ‘[YEAR]’ is not included as an object class is discussed below.

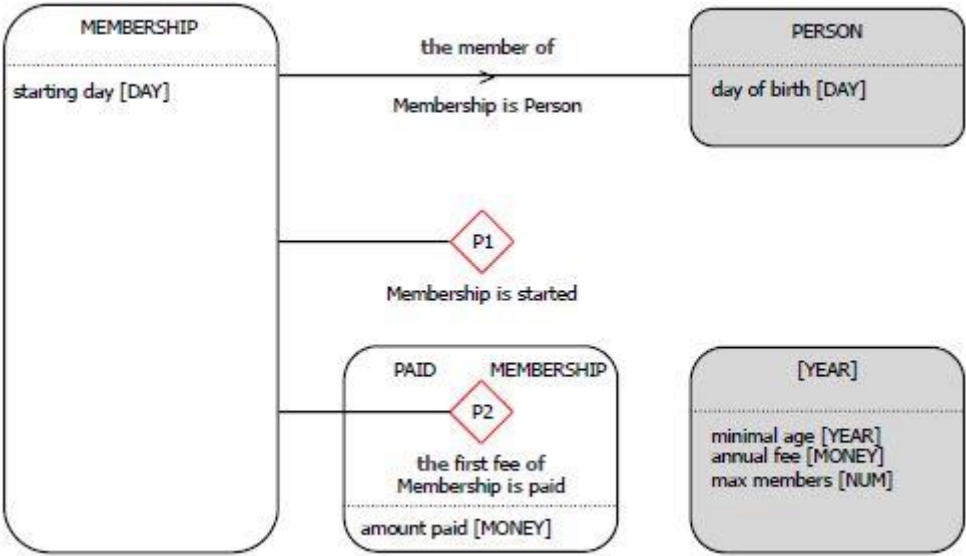


Figure 14: Object fact diagram (Perinforma, 2017)

The object classes ‘MEMBERSHIP’ and ‘PERSON’ are *core* classes. This simply indicates that these fact types cannot be derived from other fact types, but rather are declared to exist. The grey shading of the class ‘PERSON’ is used to show that it is external and thus not created

within the SOI. The line drawn between ‘MEMBERSHIP’ and ‘PERSON’ represents a *property type*.

Dietz and Mulder (2020) further elaborate on property mappings such as those represented in Figure 14. Figure 15 describes the principles of cardinality. A cardinality range consists of two numbers, the first describing the minimum cardinality, and the second the maximum. The cardinality range 1..1 in Figure 15 indicates that every rental has exactly one person as its renter. The 0..* range, on the other hand, indicates that a renter can make 0, 1, or many rentals. The ranges in Figure 15, as stated by Dietz and Mulder (2020), are the default ranges, and may be omitted.

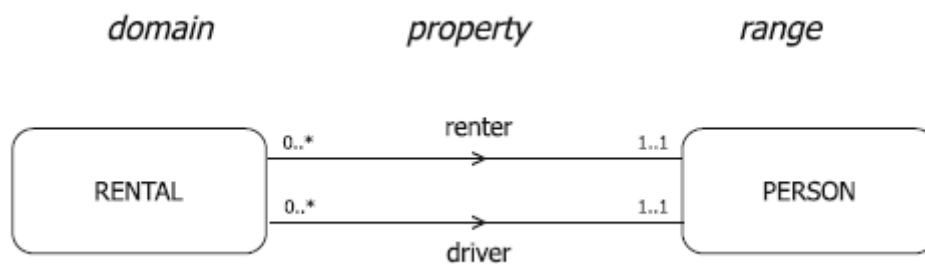


Figure 15: Property mapping (Dietz & Mulder, 2020)

The PKs P1 and P2 arise with the completion of the TKs as introduced in Figure 10. This is also why they are modelled with a diamond shape (Perinforma, 2017). Their existence therefore does not depend on the ‘MEMBERSHIP’ object class.

The ‘[YEAR]’ entity represents a fact of type *value scale* represented by the square-brackets. The use of a value scale is an elegant way to show the units in which certain facts are measured. For example, it can be seen from Figure 14 that the ‘minimal age’ of a member is recorded in years. The heading of the overall value scale, in this case ‘[YEAR]’, also communicates how frequently the values contained within it are renewed. Thus the annual fee for membership is reconsidered annually.

As explained, facts can be declared, but facts can also be derived – i.e., facts are not created but are derived from original facts. The *derived fact specifications* (DFS) are used to represent these derived facts textually. It is also possible that certain *existence laws*, sometimes referred to as *business laws or rules*, need to be defined that are outside the scope of the OFD. This can also be done textually in a structured manner in *existence law specifications* (ELS). Both the DFS and the ELS of the example can be seen in Figure 16 below.

Derived Fact Specifications

the age of Person on Day = Day minus the day of birth of Person

the number of members on Day = the cardinality of STARTED MEMBERSHIP on Day

the first fee of Membership = (12 minus the month of the starting day of Membership) times the annual fee in the year of the starting day of Membership

Business Laws

Membership is started on Day implies that Day is the first day of some Month and Month is equal to or greater than Current Month

Membership is started on Day implies that the age of the member of Membership is equal to or greater than the minimal age in the year of Day

Membership is started on Day implies that the number of members on Day is less than or equal to the max members in the year of Day

Figure 16: Derived fact specifications and existence law specifications (Perinforma, 2017)

Perinforma (2017) describes the FM and PM to be opposite sides of the same coin, since they are at the same abstraction level. Whereas the PM describes the coordination world, the FM describes the production world.

The AM

The final aspect model is the AM which, being the least abstract of the four, is the most detailed. It consists of *action rule specifications* (ARS) and *work instruction specifications* (WIS). As indicated by Dietz and Mulder (2020), the WIS is enterprise-specific, and has no generic way of representation. Dietz and Mulder (2020) further indicate that the WIS need to be viewed as detailed instructions to complete a specific production task, and that they are often, in practice, communicated via a workflow.

Perinforma (2017) describes the ARS textually, providing guidelines to actors for dealing with events they need to respond to. The sentences created in the ARS have a very specific grammatical construct so that they can be analysed in an automated manner as well.

The general form of an action rule is as follows: *<event part> <assess part> <response part>*. Regarding the presentation of the specification language, the following explanation holds. Bold elements indicate standard grammatical elements that serve to improve the readability of the sentences in the ARS. Underscored elements refer to words that have definite meaning within DEMO and can be seen as being similar to reserved words in a programming language. All other terms are organization-specific. Italics also play a role, indicating informal conditions and comments. Both the general structure and the grammatical rules of the ARS are illustrated in Figure 17, where the action rules for A1 in the ‘Volley’ example are shown.

when	membership start for new Membership is requested	(T1/rq)
with	the member of Membership is some person the payer of Membership is some person the starting day of Membership is some day	
assess	<i>justice:</i> the performer of the request is the member of Membership the addressee of the request is a membership starter <i>sincerity:</i> < no specific condition > <i>truth:</i> the starting day of Membership is the first day of some month; the age of the member of Membership on the starting day of Membership is equal to or greater than the minimal age in the year of the starting day of Membership; the number of members on the starting day of Membership is less than the max members in the year of the starting day of Membership	
if	<i>complying with the assessment is considered justifiable</i>	
then	<u>promise</u> membership start for Membership	[T1/pm]
	with the addressee of the promise is the member of Membership;	
else	<u>decline</u> membership start for Membership	[T1/dc]
	with the addressee of the decline is the member of Membership;	

Figure 17: Action rule specification (Perinforma, 2017)

Perinforma (2017) divided the action rule specification in Figure 17 into three blocks that illustrate the general form of the ARS. The top block indicates that the event for a new membership is requested. It also communicates that both the member and the payer of the membership are a person, and that the starting day is some day. On the right side of the ARS of the block, the coordination acts and facts are shown.

The middle block of the figure represents the ‘assess’ part of the ARS. The assessment is always done with a regard for *justice*, *sincerity* and *truth*. ‘Justice’ is assessed by determining the authority of the performer, ‘sincerity’ by determining whether the performer is sincere in his/her act, and ‘truth’ by validating whether certain existing facts are true. From the figure, the middle block communicates who is authorised to perform the relevant acts, how to determine whether the performers are sincere in their correspondence (no specific condition is given in the example), and which conditions are needed to be true regarding existing production facts.

The final block represents the action rules with regard to the response of the performer to a transaction. First, an informal specification is defined, stating that the response act can only happen if the assessment of the instance is considered justifiable. Then A1 can either promise or decline the request for a new membership.

This section of the dissertation has served to explain and illustrate how the different aspect models within DEMO are used to create the essential model of the enterprise. DMT, the tool created in this dissertation, incorporates the DEMOSL 3.7 specifications with extensions for the CM (see Section 1.9.6). It is in the future vision of the tool to upgrade to the newest version of DEMOSL (currently 4.5) and to incorporate all four aspect models.

1.11 Theory on BPMN

Given that an objective of this dissertation was to investigate DEMO-to-BPMN transformations, it is important to examine some theory on BPMN as well. BPMN has become an industry standard for modelling business processes. The latest version of BPMN (BPMN 2.0.2) was released in 2013 by OMG (Dumas, La Rosa, & Mendling, 2018). BPMN consists of three distinct types of diagrams: *choreography*, *conversation* and *collaboration* (Dumas et al., 2018). Of these three, the collaboration diagram is most frequently used, and is the only one included in DMT. The choreography and conversation diagrams are thus introduced in this section, but not discussed in detail.

The choreography diagram is used to model expected behaviour between participants. According to Dumas et al. (2018), the choreography diagram allows for the modelling of interactions between participants while also specifying the order in which they occur. The choreography diagram is thus modelled as events, activities, and gateways between participants (Von Rosing et al., 2014), whereas the collaboration diagram indicates the responsibility areas of participants via lanes or pools.

The conversation diagram only captures interactions between participants without specifying the order of events (Dumas et al., 2018), and is used to indicate key conversations that happen between business participants. For example, a business participant ‘Retailer’ might have a conversational link to the participant ‘Supplier’, where the conversation ‘Delivery Negotiations’ is shown (example taken from Von Rosing et al. (2014)).

As previously mentioned, the collaboration diagram is the most frequently used of the BPMN diagrams. It serves to highlight interactions between participants (modelled as pools). A pool can also be divided into lanes that represent parts of a participant. This makes sense if the participant being considered is a business or business department. For example, the business ‘ABC Construction’ might have the lanes ‘Procurement’ and ‘Finance’ (Dumas et al., 2018).

Each participant may be modelled as a white-box or black-box pool, depending on whether their responsibilities are important to the modeller. Message flows also exist between these participants, highlighting their interactions (Von Rosing et al., 2014).

The collaboration diagram represents the process flows across the created pools in terms of *events*, *activities*, *sequence flows*, and *gateways* (Dumas et al., 2018). Dumas et al. define events as things that occur instantaneously, such as ‘invoice received’ or ‘process started’.

Activities, on the other hand, represent entities that occur over a certain period of time. ‘Prepare invoice’ may be used as an example of an activity. The act of preparing an invoice, no matter how fast it occurs, is not instantaneous, and cannot be classified as an event. Sequence flows

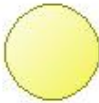



are used to show the sequential dependencies within a collaboration diagram, and indicate a moving from one event or activity to another within the process flow (Dumas et al., 2018).

Gateways represent a gating mechanism that either allows or stops the passage of tokens (Dumas et al., 2018). This can cause either a split or a merging of process flow within a process. Gateways can be either exclusive or non-exclusive. An example of an exclusive gateway could occur in the process of hiring a new employee: a candidate’s application cannot be both accepted and rejected. A gateway can thus be used in this situation to split the process flow to either the acceptance or the rejection of the candidate.

Within the collaboration diagram, the constructs have standard notations, as shown Table 8 below. These constructs are, however, only the basic building blocks for BPMN collaboration. There are several variances of these, catering for different situations. An activity, for example, can be classified as a task or as a sub-process (Dumas et al., 2018). Both can also be further broken down into different types.

Although there are many different variations in constructs, the logic behind the collaboration diagram stays the same. The BPMN 2.02 specifications document (Object Management Group, 2020) is a good resource to consult for a complete list of construct variances.

Table 8: BPMN notation (modelled in the DMT modelling tool)

<i>Construct name</i>	<i>Construct shape</i>
Event	
Activity	
Flow	
Gateway	

1.12 DEMO-BPMN transformation theory

Now that both the DEMO and the BPMN modelling languages have been discussed at greater length, the literature introduced in Sections 1.9.5 and 1.9.6 pertaining to DEMO to BPMN transformations can be discussed further.

Figueira and Aveiro (2014) indicate that, even though the AM's meta-model specification from DEMOSL was vague at the time, some concepts could be mapped from DEMO to BPMN. Some examples of these in the notation of *DEMO concept* < > *BPMN concept* are: *Transaction kind* < > *Pool*; *Actor role* < > *Lane*; *Flows and conditions* < > *Tasks*; and *Coordination acts and production facts* < > *Signals*.

Next, Mráz et al. (2017) suggest a top-down method for transforming from DEMO to BPMN, using the universal transaction pattern. Rodrigues (2017) expands on the work of Mráz et al. and suggests a methodological approach to transform from DEMO to BPMN.

However, De Vries and Bork (In review), in a newer work, argue that both Rodrigues (2017) and Mráz et al. (2017) fail to address all possible scenarios from DEMO that need to be transformed. De Vries and Bork identified four main scenarios (not comprehensive) in DEMO that can be transformed. These are all centred on the nature of the selected TK that needs to be transformed:

- *Scenario 1: Customer-initiated TKs*
- *Scenario 2: TKs initiated via a parent TK*
- *Scenario 3: Self-initiated TKs*
- *Scenario 4: A parent TK that initiates multiple part TKs*

The authors then proceed to define the transformation specifications to be used in the listed scenarios. Considering that the work by these authors provides a template to transform four main scenarios, with clear instructions on how to perform these transformations, their methodology was used for the creation of DMT.

Chapter 3

Research methodology

For the successful completion of this dissertation, the *design science research* methodology is chosen, as it fits the requirements of this project.

1.13 Literature on research methodology

Given that the thesis statement proposes the *development of a new artefact* to add value to the University of Pretoria and to other institutions, the methodologies considered were *design science research* and *action design research*.

According to Peffers, Tuunanen, Rothenberger, and Chatterjee (2007), the *design science research* methodology is used to create artefacts that solve observed problems, contribute to research, and evaluate the newly-developed artefact. The methodology then aims to communicate the results to the appropriate audiences.

Action design research, while also *developing and evaluating an artefact*, does this by intervening in an enterprise and addressing a problem. According to Sein, Henfridsson, Purao, Rossi, and Lindgren (2011), the development of the artefact and the intervention in the system is done in an iterative manner. The artefact is designed and used by the organisation; defects, mistakes or potential improvements are identified; and the artefact is redesigned. The aim of action design research, therefore, is to ensure that the produced artefact is both *relevant* and has *technological rigour*.

For the purposes of this dissertation, the design science methodology is chosen, since the artefact aims to solve a problem, but cannot be used in the context of a classroom if it is potentially incomplete and untested. This is especially true for its use in tests or exams.

1.14 Research methodology for the study

The design science research methodology used for this study follows logical steps that are described by Peffers et al. (2007). These are listed and briefly discussed below:

Step 1: Problem identification and motivation. Here, the research problem to be solved is defined and the need, or value, of a solution validated.

Step 2: Define the objectives for a solution. Given that the problem is worth solving, the next step is to determine objectives for the solution to the problem identified. The objectives for the solution are inferred from the problem identified in Step 1.

Step 3: Design and development. In this step an artefact is created that aims to meet the defined objectives. This step includes determining the artefact's desired functionality and structure, and then creating the artefact.

Step 4: Demonstration. Demonstrate how the artefact is used to address a specific instance/s of the defined problem.

Step 5: Evaluation. This step is used to indicate the extent to which the artefact supports a solution to the problem. This could include any appropriate evidence or proof that the artefact meets the defined objectives.

Step 6. Communication. Here, all information relevant to the target audience is communicated. This could include the problem, its importance, how the artefact was developed, and why it solves the problem.

Next, the theoretical steps need to be applied to this study, as follows:

Step 1: Problem identification and motivation. This is done in Chapter 1, and is achieved by first studying the problem context, then stating the problem and its significance. Section 1.9 frames the problem instance as a class-of-problems. Chapter 4 provides a further analysis of the problem.

Step 2: Define the objectives for a solution. The objectives for the artefact are stated in Sections 1.18 and 1.20. These correspond to the requirements used to evaluate the existing DEMO tools with an alteration to R1, referred to as R1*. This alteration highlights that, although it is the intention to include all four aspect models in DMT, only the CM is included in the current version.

Step 3: Design and development. The design and development of the artefact can be seen in Chapter 5. This is done by first considering the tool requirements in Chapter 4. These are the functional and non-functional requirements of the tool, implemented by DMT by extending the existing ADOxx BPMN (2.0) library also to include the OCD and TPT concepts of DEMO. Then the AdoScript code for the transformations is written and implemented (as discussed in Section 1.22).

Step 4: Demonstration. The artefact is demonstrated by using it to model some operations of a fictitious *college*, as seen in De Vries and Bork (In review). The purpose of the demonstration is to evaluate the *utility* of DMT. The demonstration can be seen in Chapter 6, and involves modelling the DEMO scenario and showcasing the semi-automated transformations to BPMN as specified by De Vries and Bork. Addressing threats to external validity, we also used the *Rent-a-Car* case from Perinforma (2017) to demonstrate the semi-automated transformations.

Step 5: Evaluation. The usability of the artefact was evaluated by involving 34 participants with some industry experience in using DMT during a modelling exercise and reporting on its usability (see Sections 1.28 and 1.29). These participants were enrolled for a postgraduate course in industrial engineering. They provided feedback on the usability of DMT by completing a SUMI questionnaire, which is a commonly used metric for evaluating the usability of software (Sauro & Lewis, 2012). The evaluation also includes a number of tests to determine whether the tool complies with the DEMO meta-model described by DEMOSL 3.7 (Dietz & Mulder, 2017) and the extensions highlighted by Mulder (2019a) (see Section 1.27).

Step 6. Communication. The information is communicated to the appropriate audiences by medium of this dissertation and by two proposed conference articles. The first of these articles has already been published at the EMMSAD conference (Gray, Bork, & De Vries, 2020). The second article has been accepted for publication at the EmpER 2020 conference (Gray & De Vries, Accepted for Publication)

1.15 *Expected rigour*

The expected rigour in respect of the validity and reliability of the evaluation results is discussed next.

To ensure that the solution proposed in this dissertation is valid, the following steps have been taken. DMT was assessed against the following five minimum requirements (elaborated in Chapter 4) and evaluated as follows:

- *R1**: DMT is demonstrated as capable to model the CM with all its objects and relationships in Section 1.23.
- *R2*: DMT is evaluated against the meta-model in Section 1.27. Here, a table is constructed testing DMT's adherence to every aspect of the meta-model via a pass/fail criterion.
- *R3*: DMT is demonstrated as being able to facilitate semi-automatic model transformations from DEMO to BPMN. Addressing threats to external validity, we used two case studies to evaluate the model transformation feature of DMT: (1) The fictitious *college* case presented by De Vries & Bork (in review); and (2) The *Rent-a-Car* case from Perinforma (2017).
- *R4*: DMT is available free of charge to educational institutions.
- *R5*: Chapter 7 highlights how usability was tested by a group of participants.

1.16 *Ethical considerations*

The only ethical consideration for this study was the use of the questionnaire in the evaluation phase. Ethical clearance for the use of the SUMI questionnaire has been granted, and each

participant has agreed to participate with the knowledge that the information will be used for this dissertation as well the relevant articles.

Chapter 4

Problem analysis and requirements analysis

To validate the problem-instance and consequently the solution-need, the existing tools (answering *Research Question 1*) were examined to determine whether any of them met the University's requirements (answering *Research Question 2*). If no tool was found that met the University's requirements, then the solution-need was considered valid, and the research project was worth completing.

1.17 Data-gathering strategy and diagnosis techniques

As mentioned above, the diagnosis technique used to validate the problem instance is to identify and *compare the existing tools with the university's minimum requirements*. Mulder (n.d.), in further research into the problem mentioned in Section 1.1, has taken steps to identify the existing modelling tools and to evaluate them against certain requirements. Mulder's research can thus be used to aid in validating the problem-instance.

Using Mulder's list of tools as a springboard, the following protocol for information gathering was followed. As primary researcher, the author of this dissertation first attempted to gain direct access to the tool in question. If this was not possible, the tool's manufacturers were contacted to verify that the information Mulder provided was up to date. Finally, if this mode failed, Mulder's PhD was used as the most recent source of information available.

In order to compare the existing tools with the university's requirements, an alignment matrix was created. An alignment matrix, according to Martinelli and Milosevic (2016), is used to establish the degree to which a project – or, in this case, a modelling tool – is aligned with the company's objectives.

As the name suggests, the alignment matrix was created to assess the modelling tool against the university's requirements. For each tool/requirement combination, a shape was used to indicate the degree to which the tool supported the requirement. The four shapes used were: filled dot (fully supports); half-filled dot (partially supports); empty dot (does not support); and star (vision). The star is used when a tool does not yet support a requirement, but it is a vision for the future development of the tool. The question mark "?" indicates that it is unclear from the preliminary research to what degree a tool supports a given requirement (see Table 9).

In order to gather data on the user-friendliness of the tools, participant experimentation was used on three of the considered tools. These are tools that the university has considered using in the past and to which it has access: Enterprise Architect, Abacus, and ModelWorld. The participant experimentation was completed by the author of this dissertation, as someone who has had no previous exposure to the tools in question. Enterprise Architect does not support

DEMO modelling. However, there is a plug-in called Plena that can be used to model DEMO within the Enterprise Architect platform. Therefore, henceforth, whenever Plena is discussed it refers to a plug-in within Enterprise Architect.

For the initial review of the existing DEMO modelling tools, the usability criteria of Nassar (2012) were used to validate a software system's usability. These criteria are listed and described below:

- *U1 Consistency*: The system needs to be consistent in its actions, so that the modeller can get used to the system without constantly having to adapt to a new way of doing things. Consistency should apply to the way in which icons and commands are displayed and used.
- *U2 User control*: The system should offer the user control in the way the model is built and run. This could include cancelling/pausing operations and undoing or redoing steps. The modeller should be able to foresee or undo errors.
- *U3 Ease of learning*: The system should be easy to learn for a new modeller. This is achieved by avoiding icons, layouts, and terms that are unfamiliar to the modeller.
- *U4 Flexibility*: The system is expected to offer different ways to accomplish the same task so that the user experiences maximum freedom. Examples include shortcut keys, different icon options, or even layout customisation.
- *U5 Error management*: The system is expected to have built-in countermeasures to prevent mistakes by displaying error messages or warning icons, or simply preventing the incorrect placement of model elements.
- *U6 Reduction of excess*: The system should avoid displaying unnecessary information or adding unnecessary functionality to the tool. The program should be functional and easy to understand.
- *U7 Visibility of system status*: The user of the system should always be aware of the status of the system. For example, if a command does not occur instantaneously, then the system should inform the user of the delay.

Each of the three modelling platforms was tested using the following approach. A simple OCD case study was chosen. This case study was then modelled on each of the platforms, with tests being done throughout to establish whether the system met the criteria described above. This is further described and carried out in Section 1.18.

1.18 Results for validating problem instance

During the development of new software application systems, the analyst needs to consider three main categories of requirements: functional requirements, non-functional requirements, and design constraints (Leffingwell, 2011). In terms of the DEMO modelling tool, functional

requirements include the inputs, outputs, functions, and features that are needed (Leffingwell, 2011) (see R1 to R3 below). The non-functional requirements (see R4 and R5 below) incorporate the qualities of a system, such as performance, cost, security, and usability. The design constraints pose restrictions on the design of the system to meet technical, business, or contractual obligations. Next, the initial requirements for the DEMO tool are presented, structured according to the first two categories – i.e., the functional and non-functional requirements. The purpose is to compare and evaluate the existing DEMO tools against the following minimum requirements defined from the perspective of a lecturer teaching DEMO:

- *R1*: The DEMO tool should be comprehensive in supporting all of the DEMO aspect models – i.e., the CM, PM, AM, and FM (refer to Figure 1).
- *R2*: The DEMO tool should support the most recent published language specification – i.e., DEMOSL 3.7 (see Dietz & Mulder, 2017) and the extensions that have been published (see Mulder, 2019a). The tool should be ready to accommodate future upgrades to the DEMO language.
- *R3*: The DEMO tool should facilitate model transformations to other modelling languages such as BPMN.
- *R4*: The DEMO tool should be available at a low cost, especially for educational purposes.
- *R5*: The DEMO tool should be usable – i.e., be user-friendly.

The evaluated software tools (with the vendor indicated in brackets) are listed below:

- *T1*: ARIS (Software AG)
- *T2*: CaseWise Modeler (CaseWise)
- *T3*: Connexio Knowledge System (Business Fundamentals)
- *T4*: DemoWorld (ForMetis)
- *T5*: EC-Mod (Delta Change Consultants)
- *T6*: Plena (Teec2)
- *T7*: ModelWorld (EssMod)
- *T8*: uSoft Studio (uSoft)
- *T9*: Visio (Microsoft)
- *T10*: Xemod (Mprise) (No longer available on the market)
- *T11*: Abacus (Avolution)
- *T12*: DMT (This study)

T11 (Abacus), developed by Avolution, has been identified and used by the University of Pretoria before. Abacus is a tool used in enterprise architecture modelling, and boasts a substantial library that incorporates many modelling languages, including DEMOSL 3.0.

However, it only supports the CM (i.e., it supports the OCD and TPT, but not the BCT), and part of the PM regarding the DEMO aspect models. Abacus is not up to date with DEMOSL 3.7, and does not support horizontal transformations between different types of models.

Also added to this list is the DMT modelling tool, built on the ADOxx platform as the main deliverable of this study. The ADOxx platform, created by OMiLAB, is freely available for academic use. It creates an environment in which users can freely extend existing libraries to suit their needs, or create new model libraries from scratch. ADOxx and the associated DMT modelling tool are discussed in greater detail later in this paper.

With the exception of Xemod and EC-Mod, the author of this dissertation has either gained access to a tool or corresponded with the relevant vendor about it. The following updates to Mulder's analysis were thus made, and are also reflected in Table 9:

- *T3: Connexio Knowledge System*: This tool can be used to model the CM, AM, and FM, and not just the CM.
- *T4: DemoWorld*: This tool has the capacity to model the CM, PM, and FM instead of just the CM.
- *T8: uSoft Studio*: This tool has been renamed from uRequire Studio to uSoft Studio by its creators.

Table 9 below shows the comparison of the above tools against the university's requirements. As can be seen, Requirement 1 (R1) is split into four columns (CM, PM, AM, FM), and shows which of these aspect models are supported by the tool. However, it does not show the degree of vertical consistency between these models.

Table 9: Comparison between tools and requirements

Requirement no ->	R1				R2	R3	R4
Aspect model	CM	PM	AM	FM			
<i>Aris</i>	●	○	○	○	○	○	●
<i>CaseWise Modeler</i>	●	●	○	●	○	○	?
<i>Connexio Knowledge System</i>	●	○	●	●	○	○	○
<i>DemoWorld</i>	●	●	○	●	○	○	●
<i>EC-Mod</i>	●	●	○	○	○	○	○
<i>Plena</i>	●	●	●	●	★	○	●
<i>ModelWorld</i>	●	●	○	●	○	○	●
<i>uSoft Studio</i>	●	○	○	○	○	○	?
<i>Visio</i>	●	○	○	●	○	○	●
<i>Xemod</i>	●	●	○	●	○	○	○
<i>Abacus</i>	●	●	○	○	○	○	●
<i>DMT (New)</i>	●	○	○	○	●	●	●

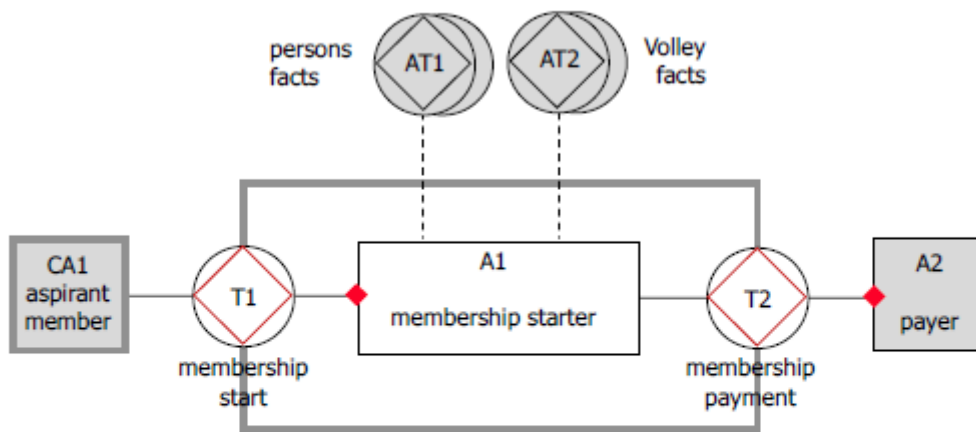
As can be seen from Table 9, only Plena includes all four aspect models that are required. Plena also has the *vision* of DEMO to be based on a sound meta-model, but this is not currently the case.

The third column shows that only the DMT tool developed for this dissertation caters for the horizontal transformation of the DEMO models into different model types. This column correlates well with the discussion in Chapter 2, where it was concluded that, although some approaches exist for transformations between DEMO to BPMN, no existing software tool facilitates the semi-automation of horizontal transformations.

Finally, regarding the column displaying the fourth requirement (low cost), a filled dot indicates that the tool listed is free, a half-filled dot indicates that the tool is not free but can be purchased by the university, and an empty dot indicates that the tool is not available to the public.

As previously mentioned, the user-friendliness of three of the tools was tested. The test was conducted as follows. The user modelled the OCD shown in Figure 18 below. This case study can be found in Perinforma (2017), and shows how memberships are started at a tennis club called Volley. The OCD was chosen because, while it is simple, it includes a number of concepts found in DEMOSL 3.7: elementary actor roles; composite actor roles (only external); transaction kinds; aggregate transaction kinds; and initiator, executor and informational links.

The image also includes, for reference, the TPT of the case study from which the OCD is derived. The TPT was not modelled in the test.



transaction kind	product kind
T1 membership start	P1 Membership is started
T2 membership payment	P2 the first fee of Membership is paid

Figure 18: OCD case study (Perinforma, 2017)

The user first attempted to model the OCD correctly, after which he tried to ‘break’ the model by inducing mistakes. The ways in which each criterion was tested are listed below.

- *U1: The user, as he builds the model, checks for inconsistencies in the way things are displayed, steps are performed, etc.*
- *U2: The user checks what control options are or are not made available to him. The obvious control options would be to cancel steps, to undo a step, or to navigate forward and backward across the completed steps.*
- *U3: To evaluate the ease of learning, it is assumed that the user has made use of the CM model before. Therefore the user will evaluate whether the technical terms, elements, and icons are familiar. It is easy to see which elements are unfamiliar when using a new interface.*
- *U4: The flexibility of a system is judged by evaluating whether there are different ways to perform the same task. In a computer tool this is done by either offering the user the freedom to customise the interface, or by offering keyboard shortcuts to perform functions. The user attempts to customise the interface, and will then attempt some common keyboard shortcuts. A list of applicable options to test is selected from Starr (2013): Ctrl+N (New); Ctrl+F (Find); and Ctrl+H (Find and replace). Also added from the user’s personal experience: Ctrl+Z (Undo); Ctrl+X (Cut); Ctrl+C (Copy); and Ctrl+V (Paste).*

- *U5: The user attempts to force mistakes onto the system to test how, and to what extent, the system manages errors. The mistakes made are: multiple actors execute a single TK; TKs are moved outside the borders of the enterprise that is modelled; execution links are placed the wrong way around; and OCD elements are placed on top of each other.*
- *U6: The user studies the interface available to him during the test, and lists all information/functionalities that are unnecessary to his task.*
- *U7: Whenever a task is being done by the system, the user notes whether or not the status is made known by the system. This could be a loading icon if a process is running, an error message if it crashes, etc.*

The results of the test are displayed in Table 10 below. Once again, a filled dot, a half-filled dot, and an empty dot are used as evaluating criteria. A filled dot indicates that, to the tester, the tool's performance fully supports the relevant user-friendliness criterion. A half-filled dot indicates that the performance partially supports the criterion; and an empty dot shows that the tool completely fails, and thus it performs at an unsatisfactory level against the criterion.

Table 10: User-friendliness test

	<i>C1</i>	<i>C2</i>	<i>C3</i>	<i>C4</i>	<i>C5</i>	<i>C6</i>	<i>C7</i>
<i>Abacus</i>	●	●	●	●	●	●	●
<i>ModelWorld</i>	●	○	●	○	●	●	○
<i>Plena</i>	●	●	○	●	●	○	●

Abacus

Abacus, as can be seen in the table, displays good results in the user-friendliness test. The user finds no obvious inconsistencies while testing the functionality of Abacus. Regarding user control, Abacus offers options for undoing and redoing actions, as well as navigating forwards and backwards within the diagram.

Abacus is also easy to learn; the icons and what they do are intuitive and largely familiar. The fact that the model elements are dragged and dropped into the diagram makes building the model easy. There is also no obvious excess in the tool interface, simplifying the learning process significantly. The only negative experience the user has in learning Abacus is that it is not immediately obvious that a diagram needs to be added to a project before the model can be built.

Abacus offers a substantial amount of flexibility in the way things can be done. To place information links, for example, the information link can be dragged and dropped from the explorer on the left of the screen, simply dragged from the appropriate elements, or specified in the toolbar on top. All the above-mentioned shortcuts also work perfectly, with the exception of Ctrl+X.

Regarding error management, Abacus has the option of validating the model against the existence rules of the underlying meta-model, displaying a table detailing all the errors identified in the model.

Finally, Abacus makes the system's status visible with a validation bar at the bottom of the screen that displays any warnings or errors. Abacus also displays which element is currently selected at the bottom left of the screen and the last action executed (saved, component added, etc.) at the bottom right of the screen. Screenshots of the completed model with and without the errors (against the underlying meta-model) made in Abacus can be found in Appendix A.

ModelWorld

ModelWorld fully supports five of the seven criteria for user-friendliness. There are no inconsistencies in the way things are displayed in this tool. ModelWorld has a very simple system for constructing the CM, which involves dragging the elements from the tool bar into the modelling space and naming them. The model tree and opened models and diagrams are also displayed to the user.

The simplicity of ModelWorld means that it is easy to learn. The elements are placed with drag-and-drop functionality from a toolbar and can be easily moved, renamed, or deleted. ModelWorld's simplicity also means that there is no unnecessary information or functionality that could cause confusion. This makes ModelWorld intuitive to use, and modellers should have no difficulty learning how to use the platform.

ModelWorld does, however, offer limited user control and flexibility. The user is unable to cancel any steps, undo any actions, or navigate forwards and backwards. Added to this, even the most basic keyboard shortcuts are not available to the user. An element cannot even be deleted by using the delete key on the keyboard; the user must right-click on the element and select the delete option from the drop-down menu.

Despite its lack of flexibility, ModelWorld provides support for mitigating errors. Elementary actor roles cannot be dragged outside of the enterprise boundary, design elements cannot be placed on top of one another, and the executor link cannot be placed the wrong way around. An elementary TK can, however, be executed by multiple ARs in ModelWorld.

Finally, ModelWorld offers no indication of the status of the system. This is made worse by the fact that there is a delay of at least a second whenever an element is placed. Not knowing what's happening on a processing level in ModelWorld can cause confusion and frustration. It is possible that the delay is because the server hosting ModelWorld is based in the Netherlands, while this test is conducted in South Africa. This delay might thus not be present in other parts of the world.

Plena

As can be seen in Table 10, Plena fully supports *Criteria 1,2,4,5* and 7, and partially supports *Criteria 3* and 6. Regarding *Criterion 1*, no apparent inconsistencies are found in the way modelling in Plena is done. The interface display and the steps followed are logical and coherent.

There is also a substantial amount of user control within Plena. The user can navigate forward and backward, and undo steps. The user is also able to control factors within the model, such as where the links are attached to the elements, or the thickness of the boundary edge.

Plena is found to be a flexible system, as there are multiple ways to perform actions, and almost all of the tested keyboard shortcuts work in Plena; the only shortcut not supported is the *Ctrl+H* keyboard combination. Plena also offers a validate option that identifies modelling errors (against the underlying meta-model) and displays them to the user.

The system's status in Plena is visible throughout the modelling process. The selected element, whether it be an element within the diagram, a tool, or a function, is displayed at the bottom of the interface. It also shows when any selected element on the interface has been created and when last it was modified.

Criteria 3 and 6 are only partially supported in Plena. The logic behind this is as follows. As a tool, Plena is easy to learn, with functionality that is familiar to the user. The difficulty arises when the Plena package has to be installed as a separate application and then imported into Enterprise Architect. How to do this is not immediately obvious; and since this initially inhibits the user from using Plena, it is deemed to affect the initial learning process.

Furthermore, because Plena is an external plugin to Enterprise Architect, some information and functionality is displayed that does not apply to DEMO models, but could apply to other types of models. Regarding the user-friendly test above, all three of the tools tested are found to be user-friendly, even though there are several improvements that could be made in this respect.

1.19 Stakeholders or users that need a solution

The stakeholders and users that need a solution are the University of Pretoria and its students. The industrial engineering department needs a tool that can be used for research as well as teaching.

This is a problem that could warrant a solution for a wider group of stakeholders. Other universities or institutions that teach and use DEMO would benefit from the tool with the functionality described. The Enterprise Engineering Institute alone identified 11 institutions globally that teach DEMO: Delft Technical University; Capgemini; Sapio; Ices; VIAgroep;

Avans+; TEEC2; Antwerp Management School; Czech Technical University; San Jose State University; and University of Madeira (Enterprise Engineering Institute, n.d.).

1.20 Requirements analysis

The requirements for DMT correspond with those described in Section 1.18 and highlighted in Table 9. For the purposes of this dissertation, as already mentioned, an initial change is made to R1 to include only the CM's OCD and TPT, with the vision to include the other aspect models in the future. The initial requirement of including only the CM is referred to as *RI**.

It is also noted again that DMT, created for this dissertation, is based on the DEMOSL 3.7 specifications with extensions, as described in Chapter 2 and not DEMOSL 4.5. This is because, when the preliminary literature review was done and the tool was built, DEMOSL 4.5 was not yet available. DEMOSL 4.5 will, however, be included in future upgrades of the tool.

Chapter 5

Design and development

The artefact created as the main deliverable for this study is a DEMO modelling tool (DMT), based on the ADOxx platform, that extends the existing BPMN library to include the CM of DEMO. In this chapter the design and development of DMT is shown. The CM elements are created within the existing BPMN library, and a script is developed to enable transformation from a DEMO organisation construction diagram to BPMN collaboration diagrams.

1.21 ADOxx and the CM

ADOxx was chosen as the platform on which to develop the above-mentioned tool, as it is free-to-use for academic purposes and, perhaps even more important, researchers are encouraged to freely develop their own tool by either starting a new library or extending an existing one. In the case of this dissertation, the author elected to extend the existing BPMN library to include the CM of DEMO, and thus allow for transformations to occur. The ADOxx platform, its standardised meta-model, and how the CM fits within the platform are now discussed.

1.21.1 ADOxx platform

The ADOxx platform is developed by OmiLab, and is described as a “development and configuration platform for implementing full-fledged modelling tools” that “supports the implementation of individual modelling languages ... or transformations” (OMiLab, n.d). In the case of this dissertation, these modelling languages are DEMO and BPMN, and the procedure of interest is *transformation*.

The ADOxx platform, in practice, consists of two parts: the Development Toolkit, into which the metamodel for a modelling language is incorporated; and the Modelling Toolkit, where the models are built by the end user. The toolkits are accessed by means of two separate applications.

Development Toolkit

The Development Toolkit is a development environment where a developer spends the bulk of his time as he develops or extends different libraries. The library screen is shown in Figure 19 below. The ADOxx Development Tool by default includes only the ADOxx 1.5 experimentation library, which contains only the ADOxx base meta-model. This meta-model is discussed further below. All other libraries must be imported from external sources to be extended or created from the default library.

In the figure below, the different iterations of DMT can also be seen. The most recent version is named ‘DEMOv1.41’ in the development toolkit, and can be downloaded from the OmiLab

website (<https://austria.omilab.org/psm/content/demo/info>). It can be seen below from the different tool versions that an iterative approach is being followed, through which the tool is constantly tested and improved.

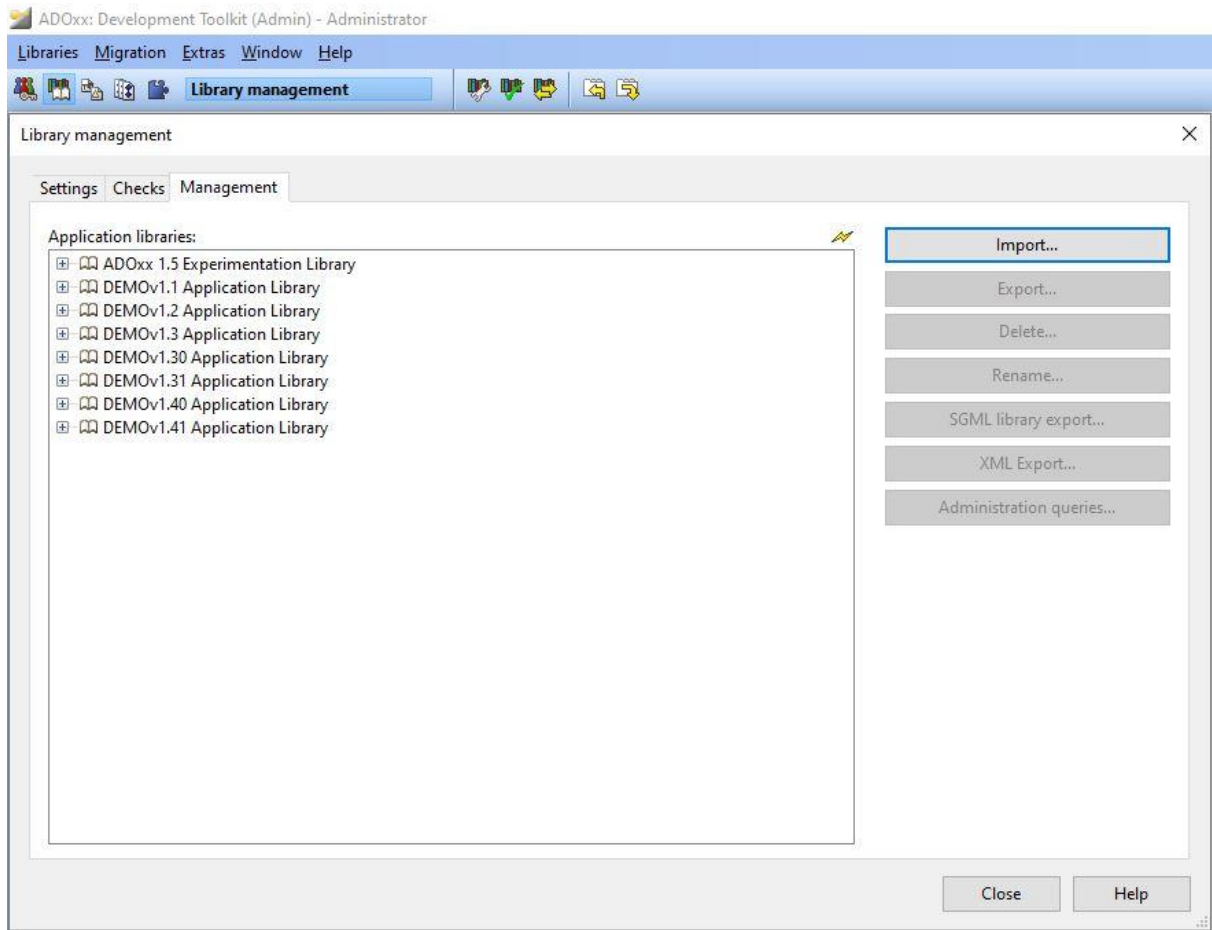


Figure 19: ADOxx development user interface

Modelling Toolkit

The ADOxx Modelling Toolkit is what the modeller will be exposed to. Figure 20 below shows an empty OCD model template. As can be seen on the left side of the screen in the ‘Explorer’ tab, the modeller can store models in different groups and sub-groups. In the example provided, ‘Models’ is the main group and ‘Sub-Group’ is the sub-group. All modelling elements are created from the ‘Modelling’ tab directly to the right of the ‘Explorer’ tab. The user simply clicks on one of these to select it, and then again on the workspace to create it.

The models can then be exported as .adl files to be backed-up or used on a different machine, or they can be exported as graphics in several different formats. ADOxx also has a host of other built-in features, as well as the capacity to create customised features. Those pertaining to this dissertation are discussed in the sections below.

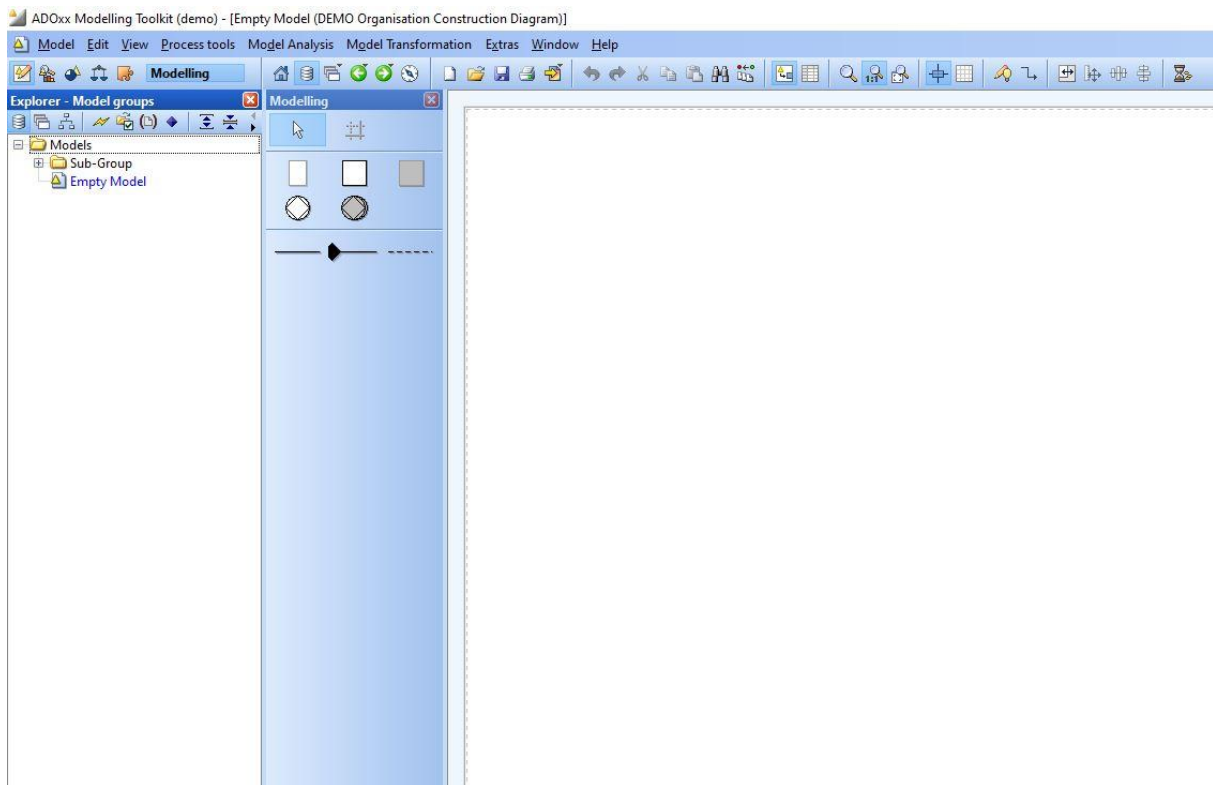


Figure 20: ADOxx Modelling Toolkit

1.21.2 ADOxx meta-model

As previously mentioned, ADOxx has a default meta-model that all modelling languages adhere to. The BPMN modelling language has already been incorporated into this meta-model. Incorporating the CM into this meta-model, while still ensuring that all models adhere to the DEMOSL 3.7 meta-model (with extensions), is the first aim of the design and development stage.

The base meta-model for ADOxx is divided into two parts, dynamic and static. For the purposes of this dissertation, only the dynamic meta-model is considered as it is extended. The static meta-model is not used because both DEMO and BPMN represent dynamic systems.

The base meta-model for the dynamic library, with an illustration of an extension, is shown below in Figure 21. The dynamic library is in turn divided into classes and relations. It can be seen from the figure that the meta-model has a hierarchical structure. Reading from the top downwards, each sub-class (child class) inherits the characteristics of the class above it (parent class). For example, the ‘_D_aggregation_’ class displays all the characteristics of the ‘_D_container_’ class. It is worth noting that the child class is not limited to the characteristics of the parent class. In the base model, the child classes also have unique characteristics that they pass on to the next class in the inheritance hierarchy. The developer of a new tool can also define characteristics that are unique to their situation in the form of class attributes.

The developer can also create brand new classes, placed as child classes of existing parent classes. These new classes then also inherit the characteristics of the parent classes. This is illustrated in Figure 21, where the new class ‘V’ is placed inside the ‘_D_aggregation_’ parent class. Here the relationship indicated by the white arrowhead signifies a new class being added. It thus inherits the characteristics of the ‘_D_aggregation_’ class.

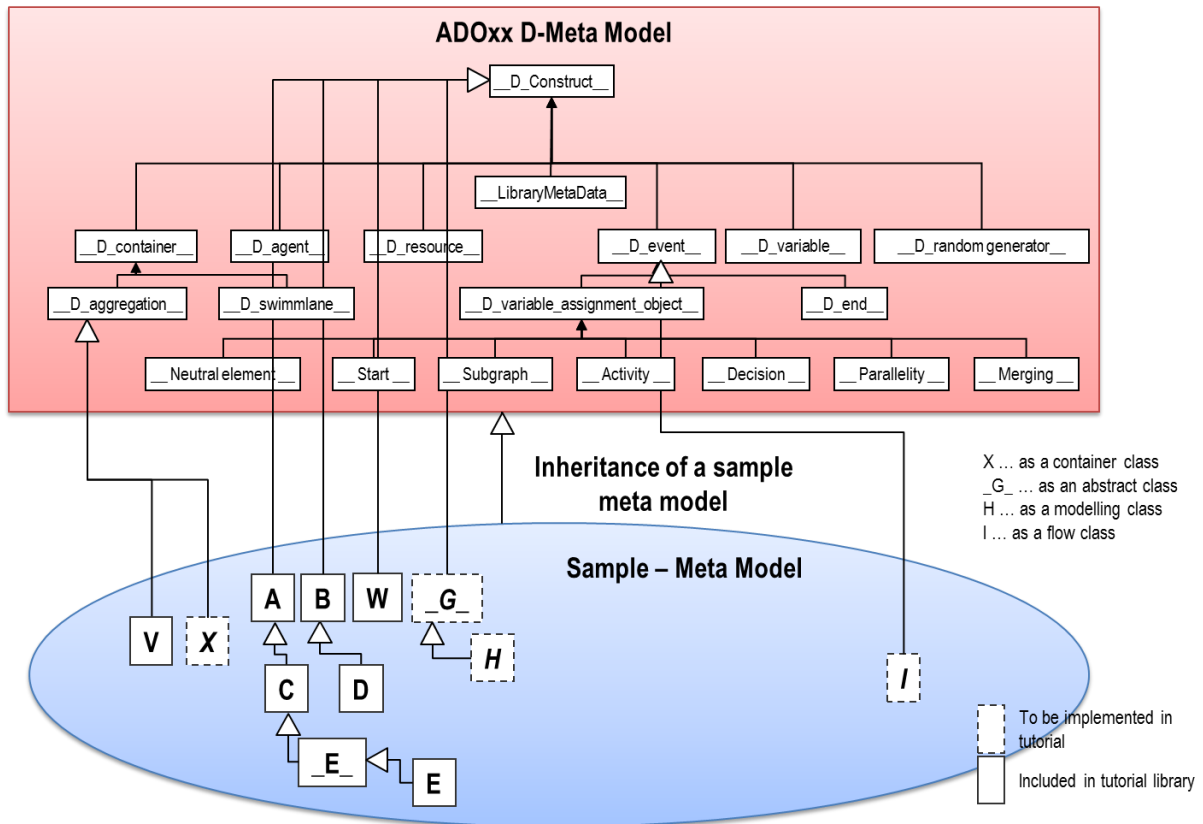


Figure 21: ADOxx Dynamic Library Meta-Model (ADOxx, n.d)

Relationship classes can then be defined between the existing classes. These relationship classes can cut across the hierarchical structure. For example, a relationship class can be created from the ‘_D_aggregation_’ class to the ‘_D_Construct_’ class. This means that the relationship can be created from any sub-class of the ‘_D_aggregation_’ class to any sub-class of the ‘_D_Construct_’ class.

These relationships can also be restricted using class cardinalities, which are used to limit or further define how a specific class deals with relationships. For example, the number of outgoing relationships allowed for a class instance, where a class instance is an object of a class, can be limited to be less than two.

For more information on the dynamic meta-model, its classes, and its relationships, the ADOxx documentation can be consulted (<https://www.adoxx.org/live/introduction-to-adoxx>).

1.21.3 CM meta-model

As mentioned in the above sections, one of the requirements of the modelling tool is that the DEMO model adhere to a sound meta-model – in this case, DEMOSL 3.7 with extensions. This meta-model is shown in Figure 22 below (Gray et al., 2020), using the general ontology specification language (GOSL). It has already been noted that the BCT component of the CM is not modelled in the first iteration of the tool, since the BCT is used to create the FM, and the FM will only be incorporated into a future version of DMT.

Given that a ‘fact kind’ and an ‘independent P-fact kind’ are part of the BCT, these entity types of the meta-model are not considered in this study.

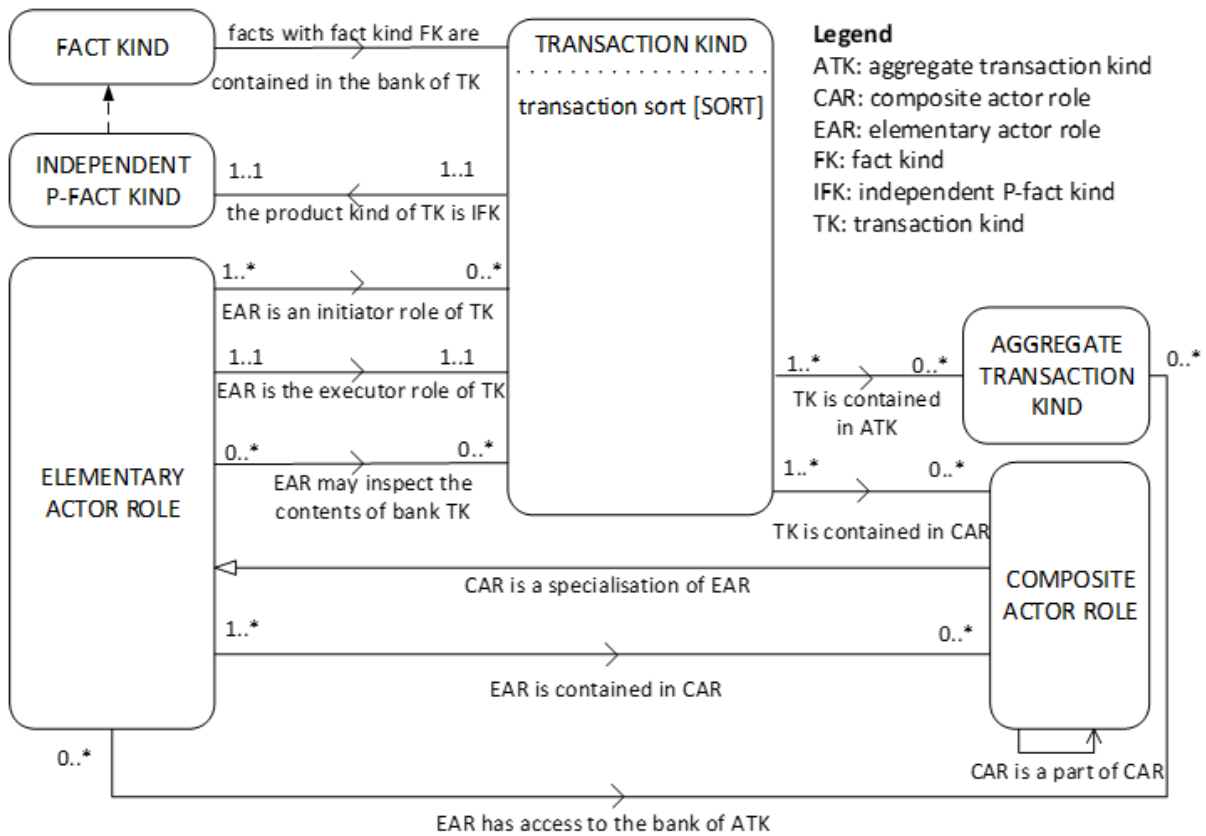


Figure 22: CM meta-model (Gray et al., 2020) based on (Dietz & Mulder, 2017) and (Mulder, 2019a)

The meta-model shown in Figure 22 presents entity types, reference laws, and cardinality laws at a schema level for a CM. Every instance of a CM has to conform to the schema level (meta-model). An example of a reference law and of its associated cardinality laws, indicated in Figure 21, is that each elementary actor role (EAR) instance must execute one, and only one, TK instance, whereas each TK instance must be executed by one EAR instance.

The next section discusses how this meta-model is incorporated into the ADOxx meta-model.

1.21.4 CM within ADOxx meta-model

To create the CM within ADOxx, the following elements need to be available for modelling, as per the CM meta-model: the CAR, EAR, TK, and ATK elements respectively.

Given that the CAR requires sub-elements to be modelled inside it (i.e., the EAR and TK elements), and that that is a specialisation of the EAR, both of these elements are placed within the ‘_D_aggregation_’ class, where the CAR is a child-class of EAR. When a non-specific reference to any of the two classes is made henceforth, the term ‘actor role’ (AR) will be used.

An important element of the CM that is not seen in the meta-model is the scope of interest (SOI). This represents the enterprise, organisation, or specific area that is being investigated, and thus requires that elements be placed inside its borders. Thus it is also placed in the ‘_D_aggregation_’ class.

Both the TK and ATK elements are placed in the ‘_D_construct_’ class, and thus need to have their characteristics defined from scratch. The reason for this is simply that none of the classes already mentioned has similar characteristics to these two. It is worth mentioning that, although the ATK has elements inside it, in practice it is never modelled within the SOI. Thus it is modelled as a black-box element on the modelling platform.

Figure 23 below shows a screenshot of how a class is added to the ADOxx library. The example class is added to the ‘_D_aggregation_’ class, which is referred to as the Superclass in the figure. In the same way, all the CM classes mentioned above are added to the ADOxx library.

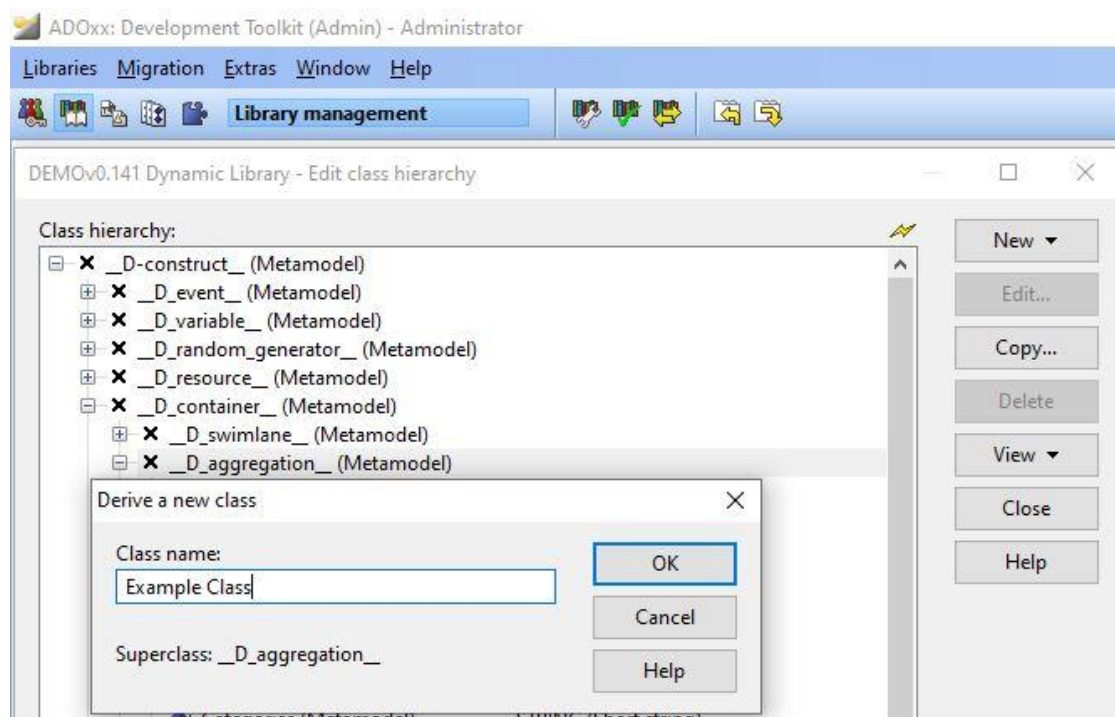


Figure 23: Adding a new class

Although these classes have now been added, they are without distinguishing features. This is where the GraphRep attribute comes into play. This attribute is added automatically to every class, and determines the shape and colour of the elements added to the modelling screen. A simple example snippet, taken from the TK class, is shown in Figure 24 below.

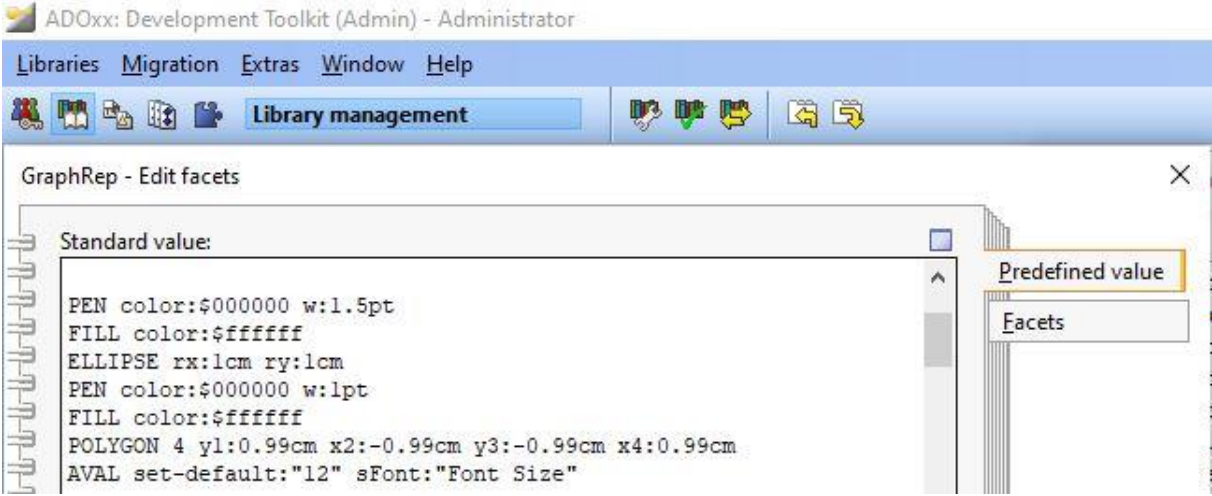


Figure 24: GraphRep of TK

As can be seen from the figure, the colour of the shape outline and its filling are defined with a colour code. The codes '\$00000' and '\$ffffff' refer to black and white respectively. Furthermore, the shape of the object is defined by a keyword and then parameters. In this case, the TK is constructed using both the 'ELLIPSE' and 'POLYGON' shapes.

The created class objects, as seen by the modeller, are shown in Figure 25 below. As can be seen in the figure, the CAR can be modelled as a black-box outside the scope entity or as a white-box inside the scope entity.

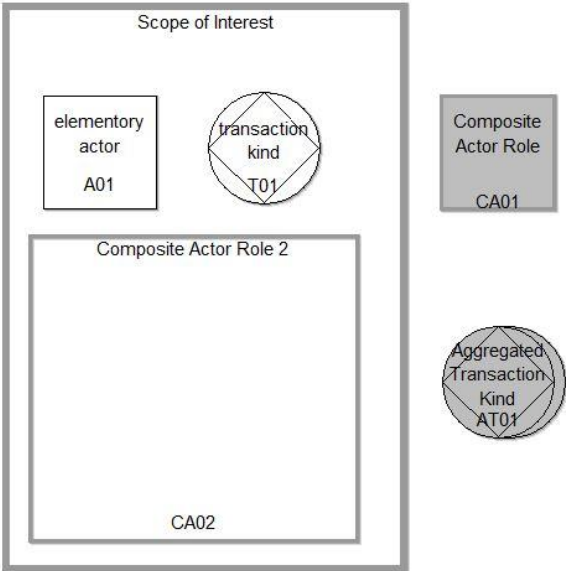


Figure 25: CM elements

1.21.5 CM meta-model compliance

Now that the class elements of the CM are available in the library, the relationships between the elements need to be created in such a way as to satisfy the CM meta-model shown in Figure 22. As found in the meta-model, four relationship types are shown in Table 11 below.

Table 11: Types of relationship in CM

<i>Relationship type</i>	<i>From Class -> To Class</i>	<i>Meta-model restriction</i>
Executor Link	AR -> TK	A one-to-one relationship applies both ways, where every AR executes one TK and every TK is executed by one AR.
Initiator Link	AR -> TK	From AR to TK a zero-to-many relationship exists, where the AR can initiate any number of TKs. For the reverse, a one-to-many relationship exists, where every TK needs to be initiated by at least one AR.
Information Link	AR -> TK/ATK SOI -> ATK	A zero-to-many relationship exists both ways for all cases. This indicates that information can be shared between zero, one, or multiple sources as needed. It should be noted that if an AR is already initiating a specific TK, information access is implied, and an additional informational link should not be possible.
Is Inside	CAR -> AR/TK SOI -> SOI/AR/TK	Here, the 'from' class references the class inside which other class elements can be placed, and the 'to' class represents the elements to be placed inside.

1.21.6 TPT creation from OCD

Finally, the TPT can be created from the OCD elements described above. The information needed to create the TPT from an OCD is: *TK name*; *TK ID*; *Product Kind name*; and *Product Kind ID*. These fields can all be filled in in the TK notebooks within the models by the user. The Notebook for the 'customer initiated tk' TK from Figure 25 is shown below in Figure 26. As can be seen, the modeller can change all the mentioned fields in the notebook.

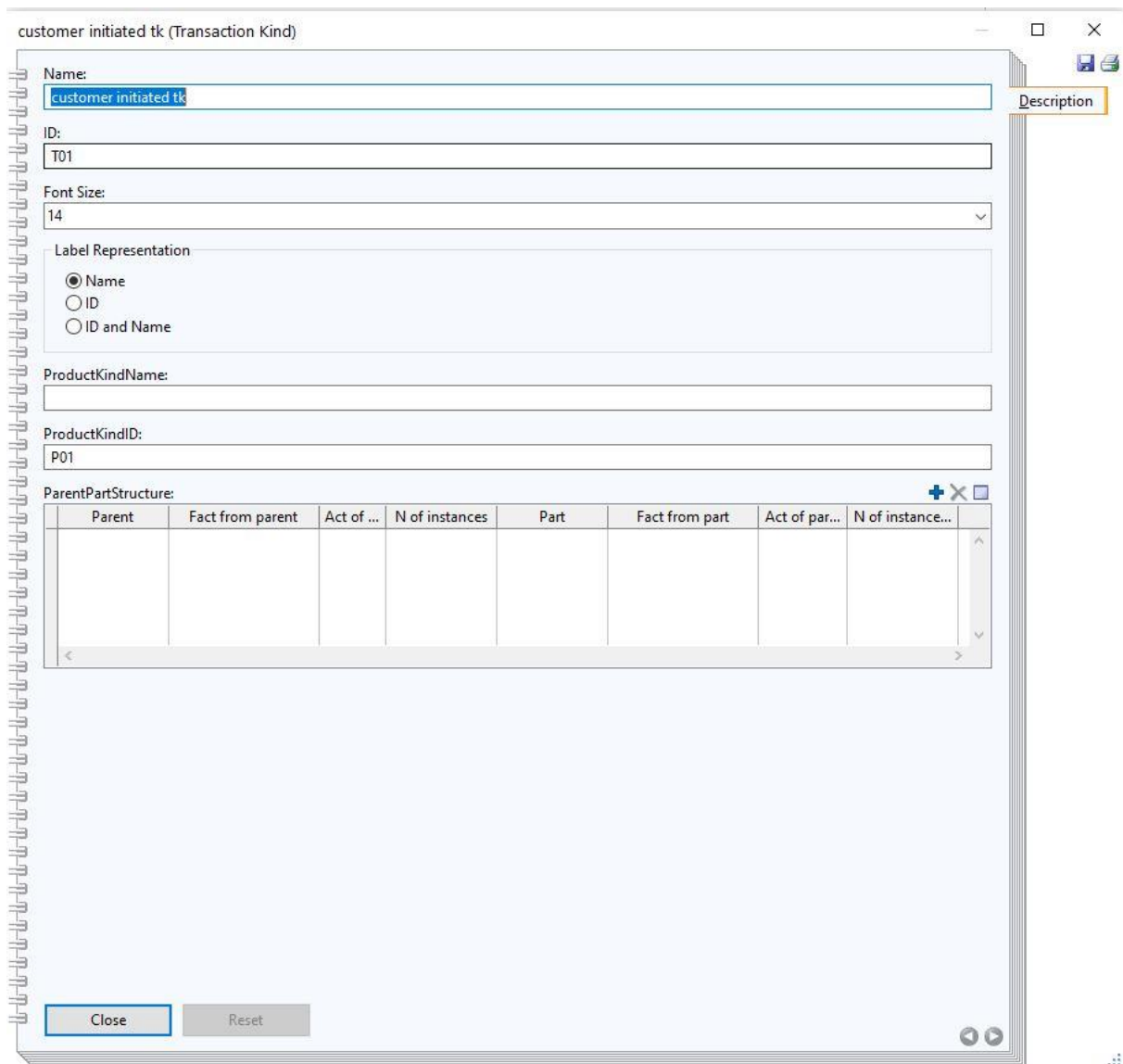


Figure 26: Notebook

To create the TPT, a custom button ‘Generate TPT’ is created and, when it is pressed, an event-trigger is used to trigger an AdoScript file. This AdoScript file then collects the necessary data from the TKs in the diagram and displays it on a table. If the product kind name is not defined, an error message pops up to remind the user to add this value to the notebook. An example of a generated TPT can be seen in Chapter 6, where the tool’s usage is demonstrated.

1.22 DEMO to BPMN transformation

For the DEMO transformation, the four scenarios discussed by De Vries and Bork (In review) are considered (see Section 1.12). Once again, they are:

- *Scenario 1: Customer-initiated TKs*
- *Scenario 2: TKs initiated via a parent TK*
- *Scenario 3: Self-initiated TKs*

- *Scenario 4: A parent TK that initiates multiple part TKs*

The transformation specifications for each of these transformation scenarios are discussed further in the article. For the creation of a tool that can successfully transform DEMO models into BPMN models, the following three-step process is established:

- *Step 1:* Automatically identify which of the four scenarios apply to the user-selected TK.
- *Step 2:* Determine where additional user input is needed, and establish how this input will be received.
- *Step 3:* Create the AdoScript files to create the BPMN models automatically from a selected TK.

How these steps are completed is discussed in the sections that follow.

1.22.1 Scenario identification

The first step in the process of performing a transformation is to automatically identify which of the four scenarios apply from the user selection. Examples of each are first modelled within DMT as shown below in Figure 27. The examples provided in the figure are not meant to be complete OCD models, but snippets of potential models where the scenarios are found.

The actual identification of these scenarios within the ADOxx platform is quite simple, and is executed using basic IF statements. The logic for each scenario identification is:

Scenario 1: The selected TK, named ‘customer initiated tk’ in the figure, is always initiated by an actor external to the SOI. Furthermore, it is executed by an internal AR that does not initiate any other TKs. With these two criteria, Scenario 1 is identified.

Scenario 2: The selected TK, named ‘tk initiated by parent’, is initiated by a parent transaction kind. Thus for this scenario the selected TK is initiated by an internal actor. This actor in turn executes another transaction – in our example, ‘parent tk scen 2’. It is worth noting that the actor executing the selected TK can be either an external or an internal actor.

Scenario 3: Here the selected TK, ‘self-initiating tk’, is initiated and executed by a single actor role. It is important that the actor role does not initiate any other TKs, as this would change the scenario into Scenario 4. With these criteria, Scenario 3 is identified.

Scenario 4: For this scenario, the selected TK, ‘parent tk scen 4’, acts as a parent TK for one or more other TKs. This scenario is identified by specifying that the executing actor, being an internal actor, must initiate at least one other TK. As mentioned in the Scenario 3 identification, the selected TK may also be a self-initiating transaction kind. It may also be the case that the part TKs are also parent TKs to other TKs.

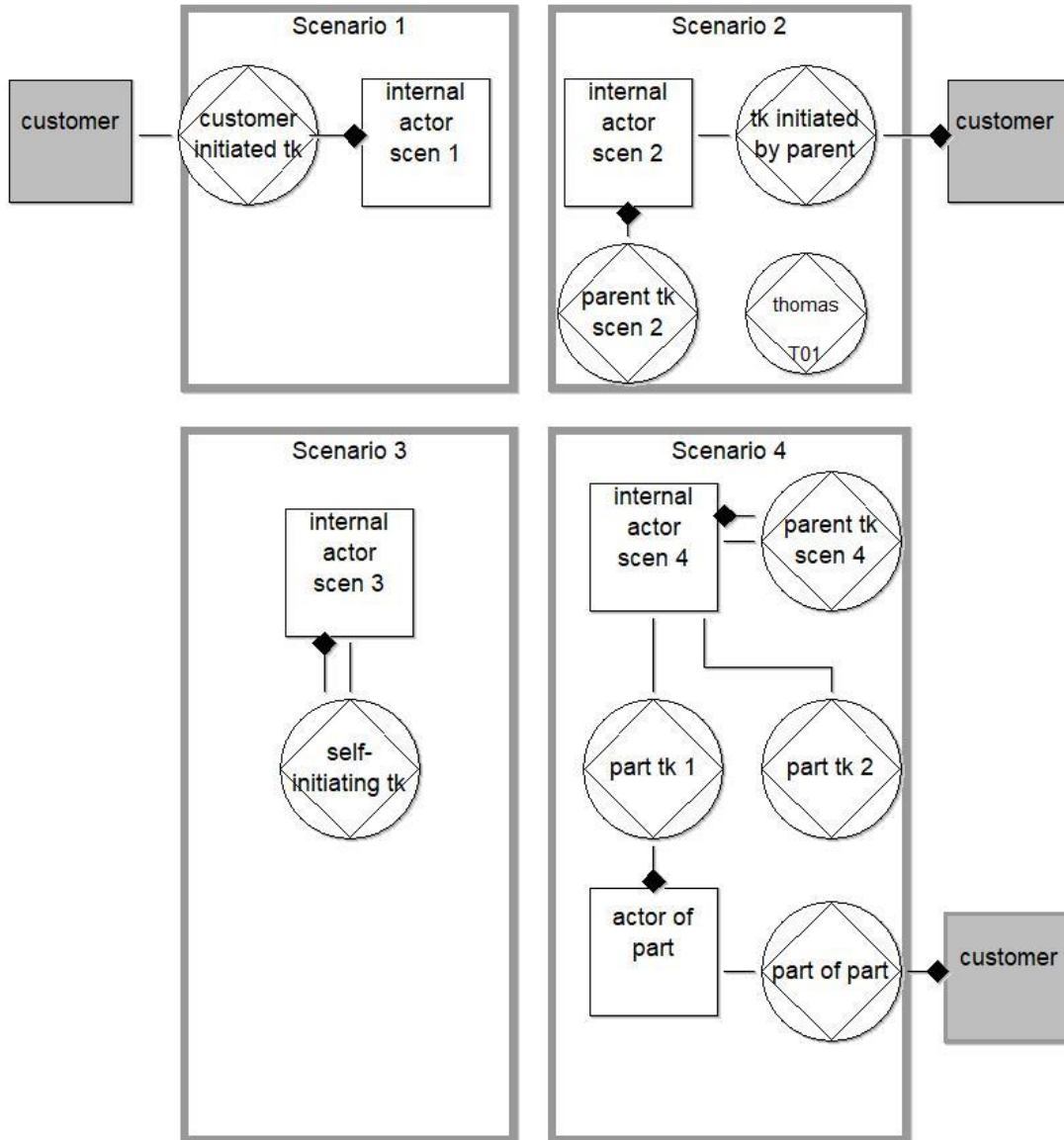


Figure 27: The four scenarios

Using the above logic, simple IF statements within the ADOxx platform are established to identify the scenarios and ensure that the correct AdoScript document is selected by the tool (see Section 4.3.3).

1.22.2 Handling of user input

Given that DMT can now identify the different scenarios, the next step is to identify which user inputs are needed and how these inputs are received. The first universal user input needed is the TK selection for the relevant model. This input determines which TK will be transformed. In DMT, a list is generated displaying all TKs contained in the model, enabling the user to pick one.

As stated by De Vries and Bork (In review), only the second and fourth scenarios need further input from the user. The user interface envisioned in this article is used as a framework for the interface used in DMT. The input table is shown in Figure 28 below. This table is editable within the user-selected TK’s notebook (see Figure 26).

ParentPartStructure:							
from parent	prereq parent fact	act of part	# of part instances	to part	prereq part fact	act of parent	# of prereq part instances

Figure 28: Parent part structure

These fields are discussed at length by De Vries and Bork (In review). It has thus been decided that, for the purposes of this dissertation, only how the user interacts with them will be explained.

from parent: Here the user specifies the parent TK from a list of all TKs.

prereq parent fact: This refers to the prerequisite fact from the parent in the parent-to-part transition, and is based on the basic transaction pattern. The user therefore has the choice of ‘requested’; ‘promised’; ‘stated’; and ‘accepted’.

act of part: This refers to the part’s act in the parent-to-part transition, and can have the values ‘request’; ‘promise’; ‘state’; and ‘accept’.

of part instances: This refers to the number of instances of the parent-to-part transition, with the following values available for selection: ‘0..1’ (zero-to-one); ‘1..1’ (one-to-one); ‘0..*’ (zero-to-many); and ‘1..*’ (one-to-many).

To part: Here the user specifies the part TK from a list of all TKs.

Prereq part fact: This refers to the prerequisite fact of the part in the part-to-parent transition, and contains the fields ‘requested’; ‘promised’; ‘stated’; and ‘accepted’.

act of parent: Like the *act of part*, this refers to the parent’s act in the part-to-parent transition, and so can have the values of ‘request’; ‘promise’; ‘state’; and ‘accept’.

of prereq part instances: This refers to the number of instances of the part-to-parent transition, and contains the same options as the *# of part instances* field above.

For Scenario 4 this table can have multiple rows, as it is used to specify all parent-to-part relationships that are relevant to the BPMN generation. Examples of how this table is filled in regarding Scenarios 2 and 4 can be seen in Chapter 6, where all four scenarios are demonstrated.

1.22.3 BPMN transformation

The transformation specifications in De Vries and Bork (In review) are not discussed in this dissertation. Rather, the semi-automatic implementation of these transformations within the ADOxx platform is elaborated on.

For the transformation, like the TPT generation, a button is displayed to the user that triggers an AdoScript file. This file generates a list that displays all the available TKs within the DEMO model. The user then selects the TK he wants to transform. Next, DMT identifies, according to the criteria in Section 1.22.1, which of the four scenarios apply to the selected transaction kind. Finally, an additional AdoScript file, corresponding to the appropriate scenario, is run to generate the BPMN model.

For Scenarios 1, 2 and 3, the transformed BPMN has a set arrangement of elements, with their names being altered according to the specific situation. To perform this transformation, the developer uses an attribute applicable to all BPMN elements called 'Position'. This attribute determines where on the worksheet the elements are placed.

It is then easy, using AdoScript notation, simply to create the new model and create the appropriate class and relationship elements. The attribute 'Position' for each of these is then altered to ensure that the placement of all objects is correct.

Finally, the naming of the new BPMN elements is determined from the OCD. This is simply done by retrieving the values from the OCD and changing the 'Name' attribute within the AdoScript file. For example, the BPMN pools are named according to the executing and initiating actors within the OCD.

For Scenario 4, the BPMN diagram has the possibility to become more complex, as the TK selected by the user can be the parent to multiple parts. These parent-to-part relationships can also be arranged differently within a hierarchy. For example, Figure 29 below shows two possibilities of parent-to-part relationship for the Scenario 4 example in Figure 27.

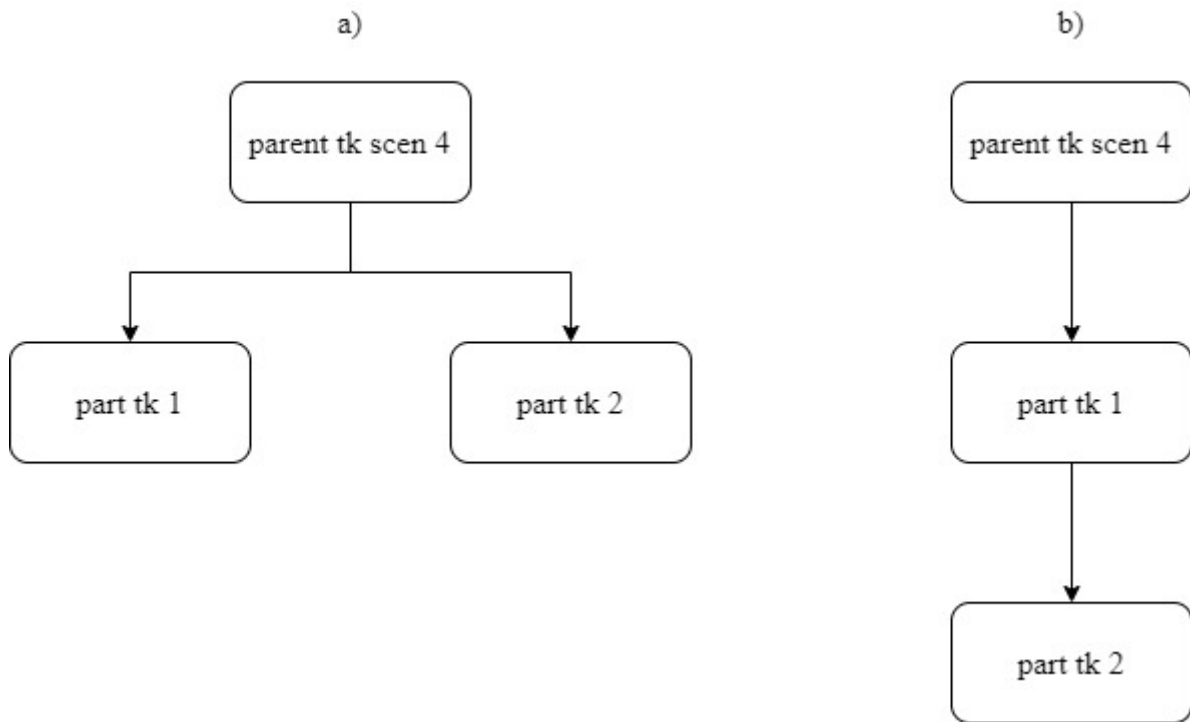


Figure 29: a) Hierarchy 1 b) Hierarchy 2

In Figure 29: a), the parent TK is parent to both part TKs, meaning that there are two distinct part TKs occurring independently from each other. For Figure 29: b), the parent TK is parent to one of the part TKs which, in turn is parent to the other part TK. This means that, although there are two part TKs, for ‘part tk 2’ to occur, ‘part tk 1’ must first be executed. This difference in hierarchy changes the format of the BPMN diagram.

It is also possible, as mentioned in Section 1.22.1, that a part TK is executed by an AR that initiates another TK creating a new parent-to-part relationship. This is depicted in Figure 27 by the new parent TK, ‘part tk 1’, and its part TK, ‘part of part’. This is thus a situation that needs to be considered in the BPMN diagram.

Given the above information, it is evident that Scenario 4 cannot be transformed in the same way as Scenarios 1,2, and 3. The logic of the other three scenarios can, however, be extended to accommodate Scenario 4. For example, the size of the main pool in the BPMN can be changed according to the number of parent-to-part relationships (rows in Figure 28) that are present. The rest of the pools are then moved to accommodate this change.

The parent-to-part hierarchy is specified in the same table shown in Figure 28. How the hierarchies for both situations in Figure 29 are specified by the user can be seen in Figure 30 a) and b) below. The parent-to-part relationship between “part tk 1’ and ‘part of part’ is also shown in these tables.

a)

ParentPartStructure: + X □								
	from parent	prereq parent fact	act of part	# of part instances	to part	prereq part fact	act of parent	# of prereq part instances
1	parent tk scen 4				part tk 1 (Tr			
2	parent tk scen 4				part tk 2 (Tr			
3	part tk 1 (Transa				part of part			

b)

ParentPartStructure: + X □								
	from parent	prereq parent fact	act of part	# of part instances	to part	prereq part fact	act of parent	# of prereq part instances
1	parent tk scen 4				part tk 1 (Tr			
2	part tk 1 (Transa				part tk 2 (Tr			
3	part tk 1 (Transa				part of part			

Figure 30: a) Hierarchy 1 b) Hierarchy 2

As shown in Figure 30, the user simply specifies which TK acts as the parent for each row. The AdoScript file is then coded in such a way to recognise the hierarchy and create the BPMN diagram accordingly.

Now that the logic governing the automatic transformation has been discussed, two case studies is demonstrated in the next chapter.

Chapter 6

DMT demonstration

To demonstrate how to model using the CM and then perform a transformation, two case studies are considered. The first case study is the same fictitious college used by De Vries and Bork (In review). This was used not only because it demonstrates the use of all possible elements and relations within the OCD, but also because it includes instances of the four scenarios (see Section 1.22.1).

The second case is that of Rent-a-Car, taken from Perinforma (2017) with some modifications to accommodate the four scenarios. The Rent-a-Car case is used because it demonstrates not only the OCD, but also the transformation of a case already used within the literature. It is in this case that the attempted transformations of Scenarios 2 and 4 expose some flaws in this initial version of DMT. These are further discussed in the sections below.

In Sections 1.23 and 1.24, both the OCD and TPT functionality and the transformation capability of the tool are demonstrated by recreating the first case study in the ADOxx tool. Sections 1.25 and 6.4 demonstrates the same functionality with the Rent-a-Car case with the exception of the transformations related to Scenario 2 and 4 which prove problematic with DMT.

1.23 *CM demonstration for the college case*

The OCD of the college, as created in the ADOxx tool, can be seen below in Figure 31. Here, some of the operations at the college are modelled. The figure includes an SOI (some operations at college); two overlapping CARs (College and Controller); and a myriad of EARs and TKs seen within the scope of interest. Outside the SOI, another CAR (*student*), a few ATKs, and two CARs (*project sponsor* and *hr of project sponsor*) are placed.

A phenomenon found in the OCD in Figure 31 that was not discussed previously is the CAR modelled as a white-box element. Two such examples (College and Controller) are present in Figure 31. As can be seen, the CARs both encapsulate EARs and TKs. It is further demonstrated in the OCD that a CAR can include elements that fall both within and outside the SOI.

Next, between the elements, all three relation types are created. It can be seen from the figure that both initiator and executor links can be placed between ARs and TKs. Furthermore, as in the relationship between the AR 'module revisor' and the TK 'module revision', a TK can be initiated and executed by the same AR, thus creating a self-initiating transaction.

Finally, the information links within the figure are placed. As demonstrated, these are placed between ARs and TKs, ARs and ATKs, or ATKs and the SOI. It is also possible for multiple information links to be connected to a modelling element.

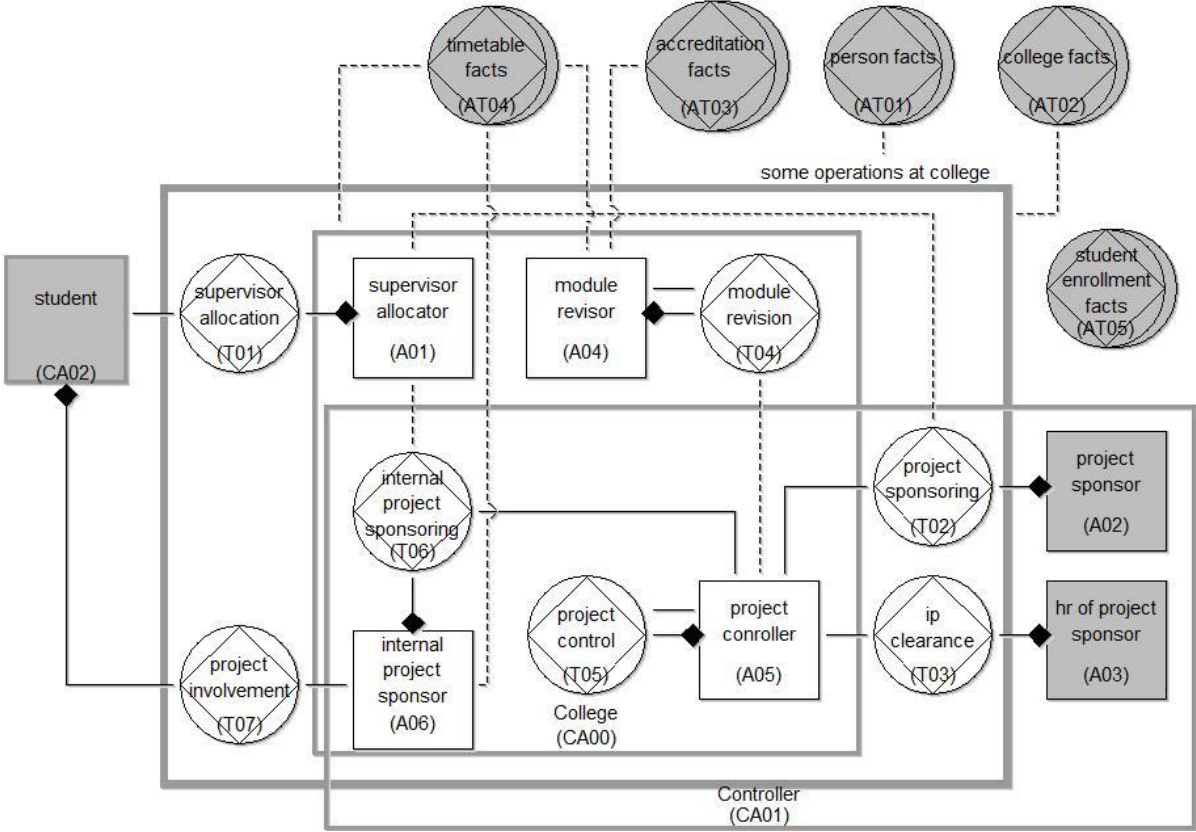


Figure 31: OCD example of college

From the OCD above, once the PK name for each TK has been specified by the modeller (see Figure 26), a TPT can be created. This TPT is shown in Figure 32, and displays a *transaction ID*, *transaction kind*, *product ID*, and *product kind* for each TK in the OCD.

transaction ID	transaction kind	product ID	product kind
(T01)	supervisor allocation	(P01)	Supervisor-allocation is started
(T02)	project sponsoring	(P02)	the sponsorship of Project is done
(T03)	ip clearance	(P03)	the ip-clearance for Project is obtained
(T04)	module revision	(P04)	the revision of Module is done
(T05)	project control	(P05)	the project-control for Period is done
(T06)	internal project sponsoring	(P06)	the internal sponsorship of Project is done
(T07)	project involvement	(P07)	Project-involvement is started

Figure 32: TPT example of college

1.24 *OCD to BPMN transformations for the college case*

The four scenarios are represented in the OCD by the following TKs:

- *Scenario 1: supervisor allocation* – Here the user-selected TK is initiated by an AR outside the SOI and executed by an AR within the SOI. The AR does not initiate another TK.
- *Scenario 2: project involvement* – The user-selected TK forms part of another TK (i.e., internal project sponsoring).
- *Scenario 3: module revision* – The user-selected TK is self-initiating. Also, the AR (module reviser) does not initiate another TK.
- *Scenario 4: project control* – This TK plays the role of a parent for several part TKs.

These are thus the TKs that will be used to demonstrate the transformations from OCD to BPMN.

1.24.1 Scenario 1 transformation for college

Scenario 1, as previously mentioned, only needs the user to specify which TK is of interest in the transformation. The BPMN construct automatically generated from the *supervisor allocation* TK is shown in Figure 33.

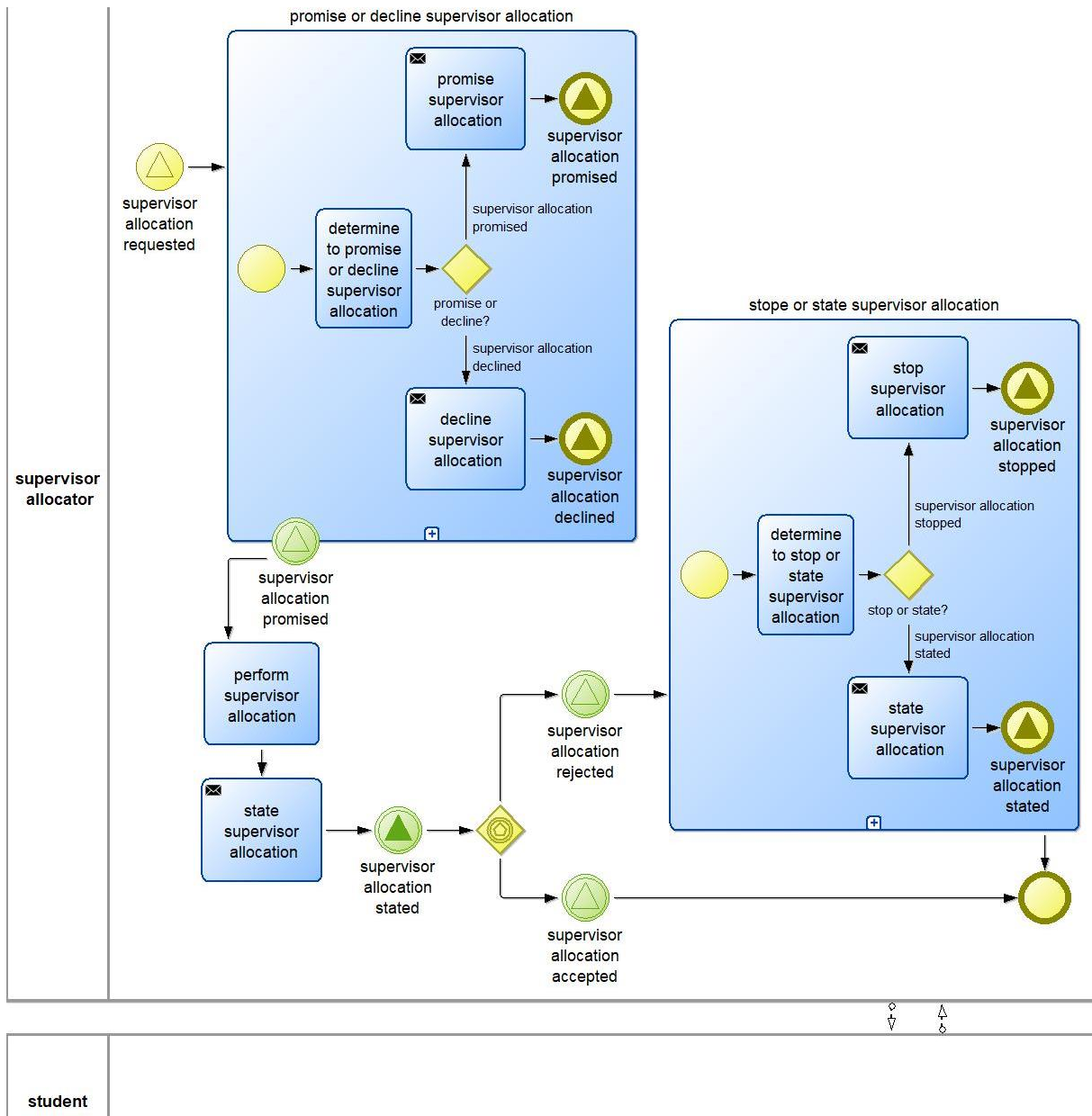


Figure 33: Scenario 1 transformation for college

As can be seen, the bulk of the BPMN diagram is generated in the *supervisor allocator* pool. This corresponds to the initiating AR in the DEMO model. The *student* pool, on the other hand, remains empty. This is because the *student* AR in the OCD is outside the SOI, and is thus modelled in the BPMN as a black-box element.

The BPMN concepts are created from the OCD model using the transformation specifications, as stated in De Vries and Bork (In review).

1.24.2 Scenario 2 transformation for college

For Scenario 2, the user needs to provide input regarding the parent-to-part relationship (see Figure 28). The user input used to demonstrate the transformation is shown in Figure 34 below. The BPMN diagram thus generated can be seen in Figure 35.

from parent	prereq parent fact	act of part	# of part instances	to part	prereq part fact	act of parent	# of prereq part instances
internal project sponsoring	stated	request	0..*	project involvement	accepted	accept	0..*

Figure 34: Parent part structure scenario 2 for college

If parent-part-structure information is incomplete, a pop-up error appears instructing the user to navigate to the TK's notepad and fill in the table. Once the table is complete, the user can continue to perform the transformation. The following BPMN diagram is generated.

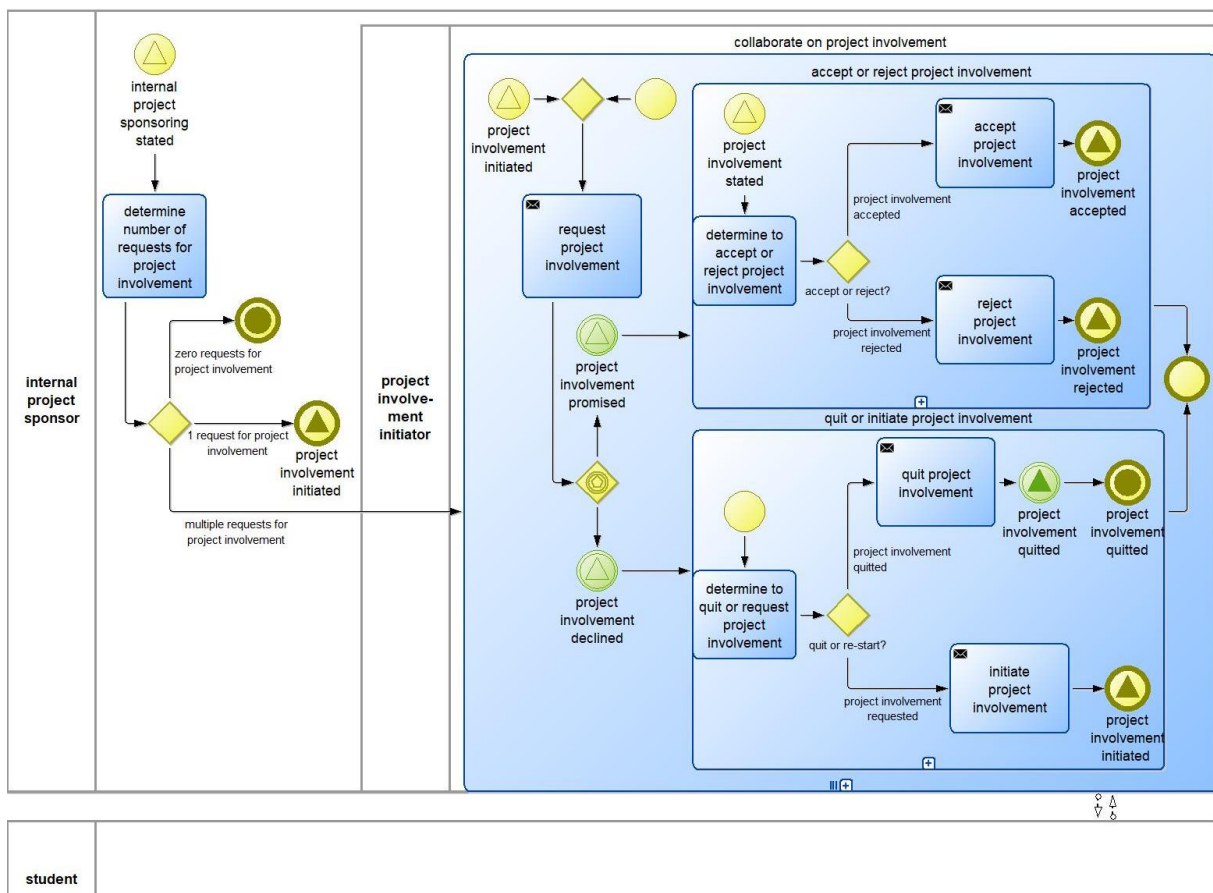


Figure 35: Scenario 2 transformation for college

As can be seen, the parent can make zero, one, or many *requests* for product involvement. If the table specifying the parent-to-part relationship had been filled in differently, the BPMN diagram would be different. For example, the parent could make only one *request* for project involvement. The table is thus used to alter the BPMN to the user's specifications, and plays an important role in the transformation.

1.24.3 Scenario 3 transformation for college

The BPMN diagram generated from the OCD is shown in Figure 36 below. As previously mentioned, the *module revision* TK is of interest here. Furthermore, only one pool is generated in the BPMN, given that this is a self-initiating TK. For the purposes of this demonstration, the *module revision* is initiated yearly. This information is provided by the user by simply renaming the start-event.

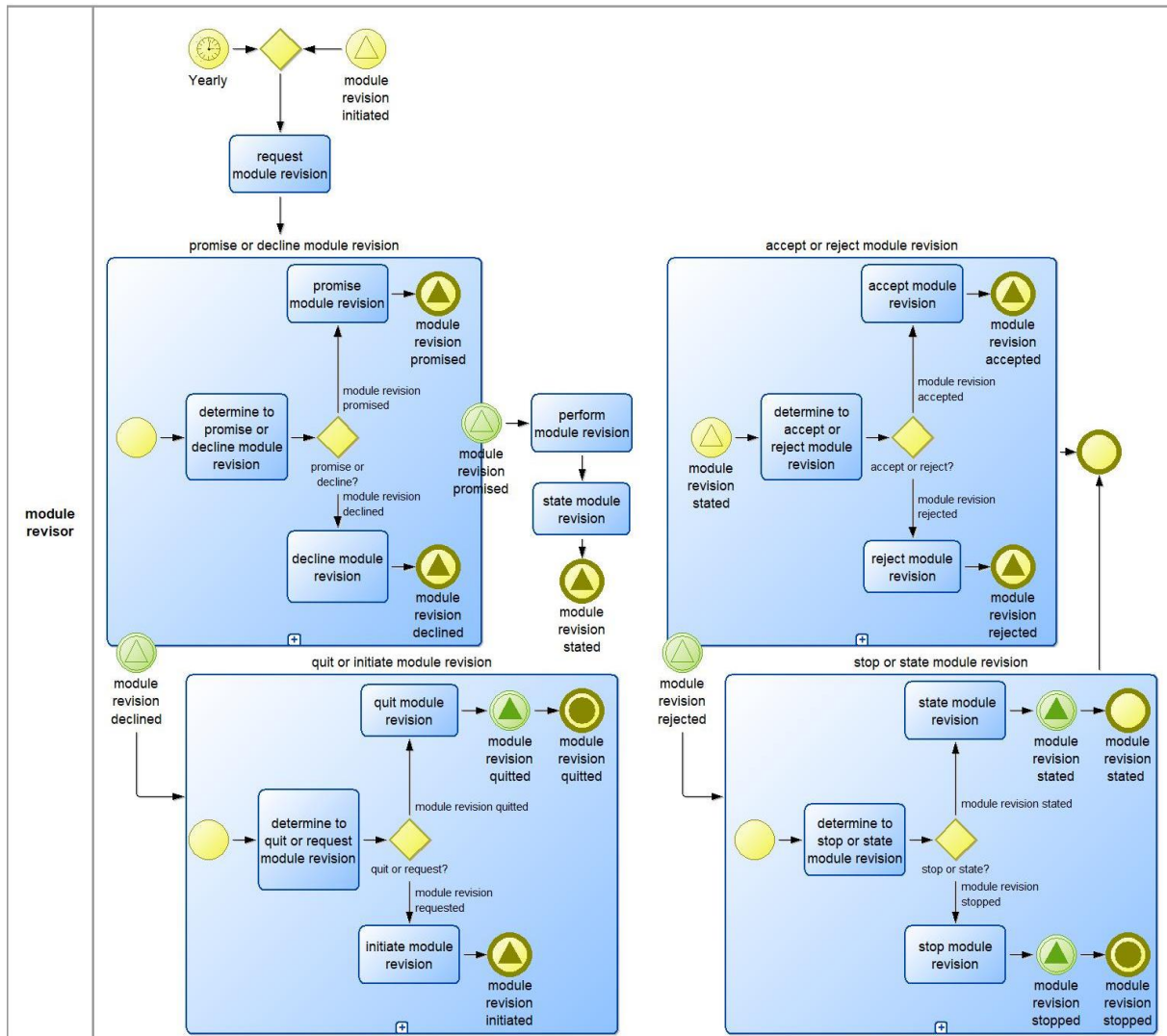


Figure 36: Scenario 3 transformation for college

This BPMN encompasses the complete transaction pattern, where the *module revision* can be requested either on a yearly basis or re-initiating the instance when an instance of module revision was declined. The module reviser then needs to promise, perform, state, and accept the module revision for a positive outcome to the process.

1.24.4 Scenario 4 transformation for college

In this final transformation, the *project control* TK is selected. Figure 37 shows the parent-to-part structure of the demonstration case. As can be seen, the TK considered, *project control*, is parent to both *project sponsoring* and *internal project sponsoring*. *Project sponsoring*, in turn, is parent to *ip clearance*, and *internal project sponsoring* to *project involvement*.

from parent	prereq parent fact	act of part	# of part instances	to part	prereq part fact	act of parent	# of prereq part instances
project control (Tra	stated	state	0..*	project sponsoring	accepted	accept	0..*
project sponsoring	stated	state	1..1	ip clearance (Trans	accepted	accept	1..1
project control (Tra	stated	request	0..*	internal project spc	accepted	request	0..*
internal project spo	accepted	promise	0..*	project involveme	accepted	accept	1..*

Figure 37: Parent part structure scenario 4

The resultant transformation of this hierarchy is shown in Figure 40 below. Once again, all relevant actors appear as pools, and the standard transaction pattern is still followed. For this demonstration the user has not yet specified the periodicity of the process. For simplicity's sake, this scenario creates a range of collapsed sub-processes; two of note, namely *collaborate on project sponsoring* and *collaborate on internal project sponsoring*, are shown in Figure 38 and Figure 39 respectively.

These are shown to demonstrate DMT's ability to model the sub-processes regarding the hierarchy specified in Figure 37. Figure 38 illustrates not only that *project sponsoring* is parent to *ip clearance*, but also that *project sponsoring* is parent to *ip clearance*. The BPMN diagram thus reflects the process dependencies that a hierarchy of TKs create.

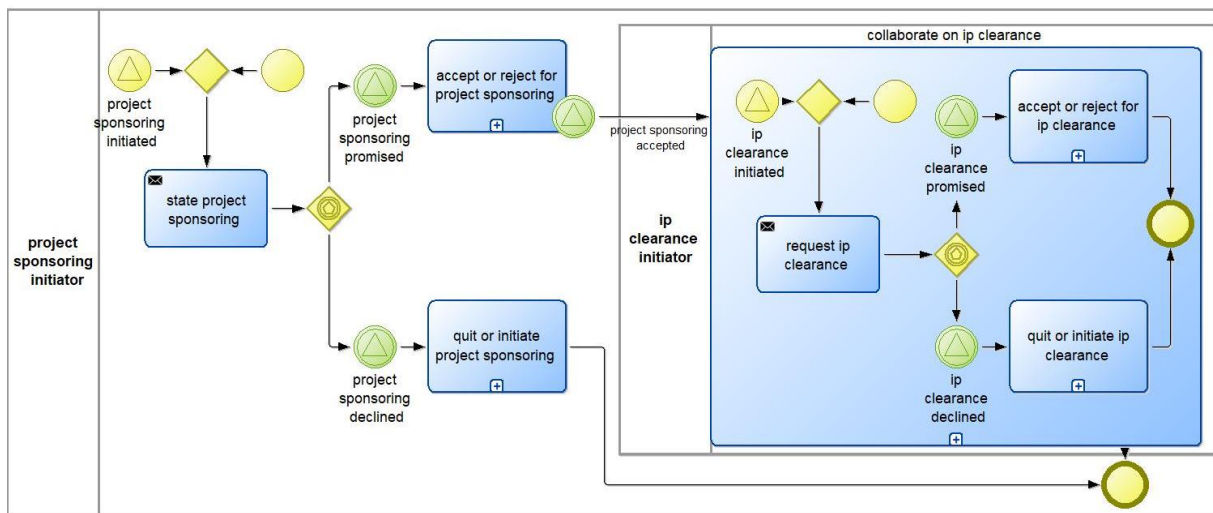


Figure 38: Collaborate on project sponsoring

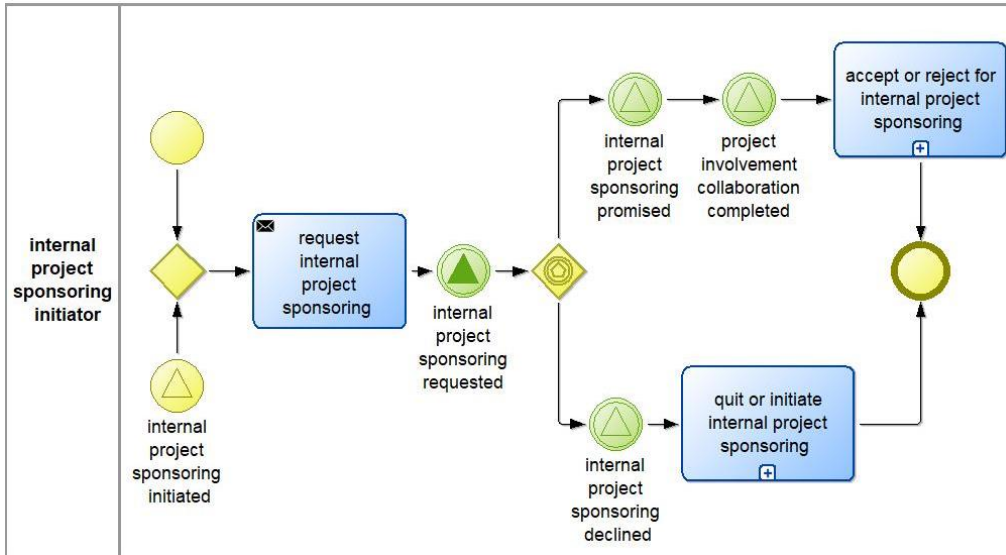


Figure 39: Collaborate on internal project sponsoring

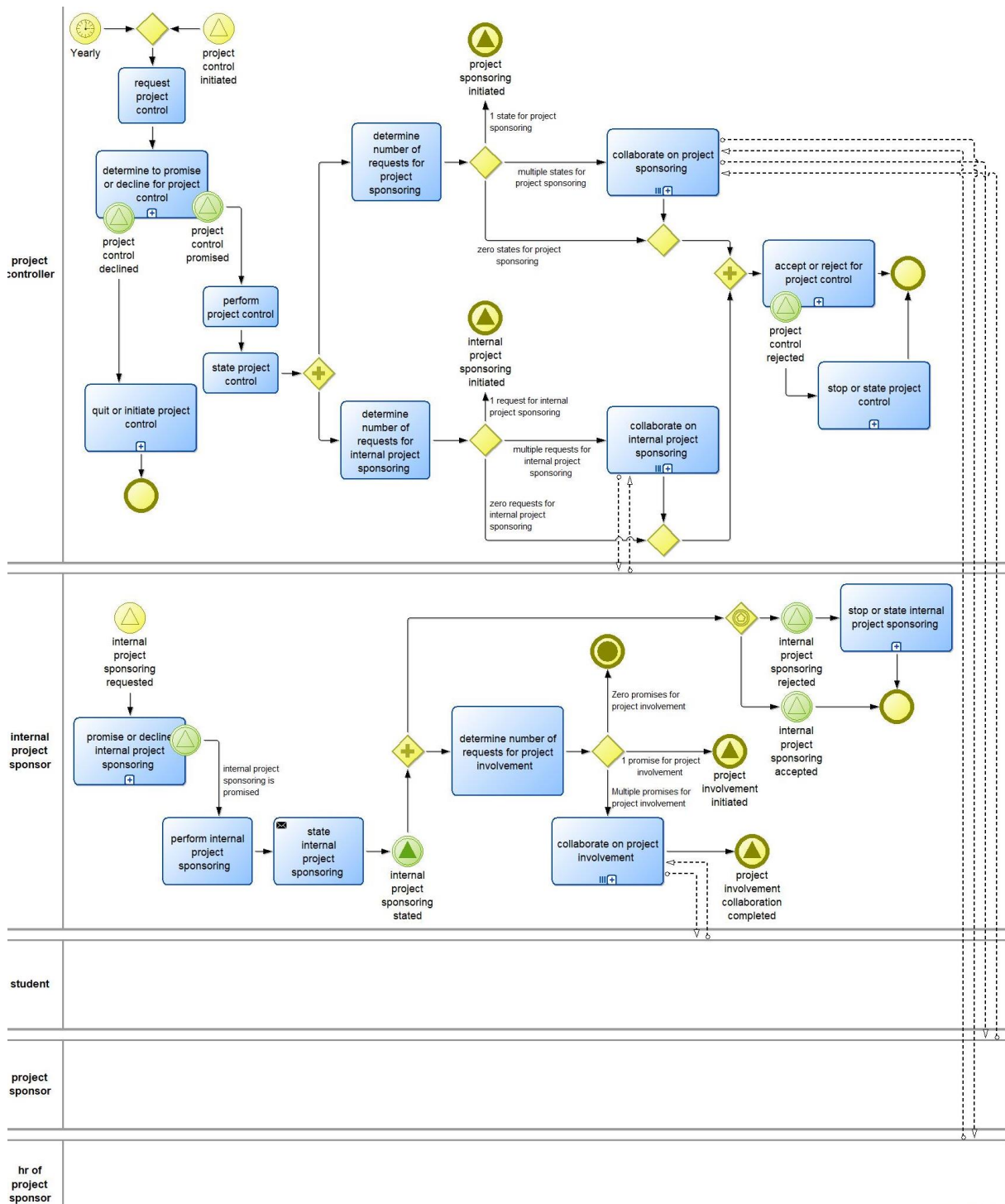


Figure 40: Scenario 4 transformation for college

1.25 CM demonstration for the Rent-a-Car case

The next case to be demonstrated is the Rent-a-Car case. Given that the OCD and the TPT for the case are already explained by Perinforma (2017), only the implementation of these are shown in Figure 41 and Figure 42. It is worth noting that some additions have been made to account for the different scenarios' state. These are discussed further below.

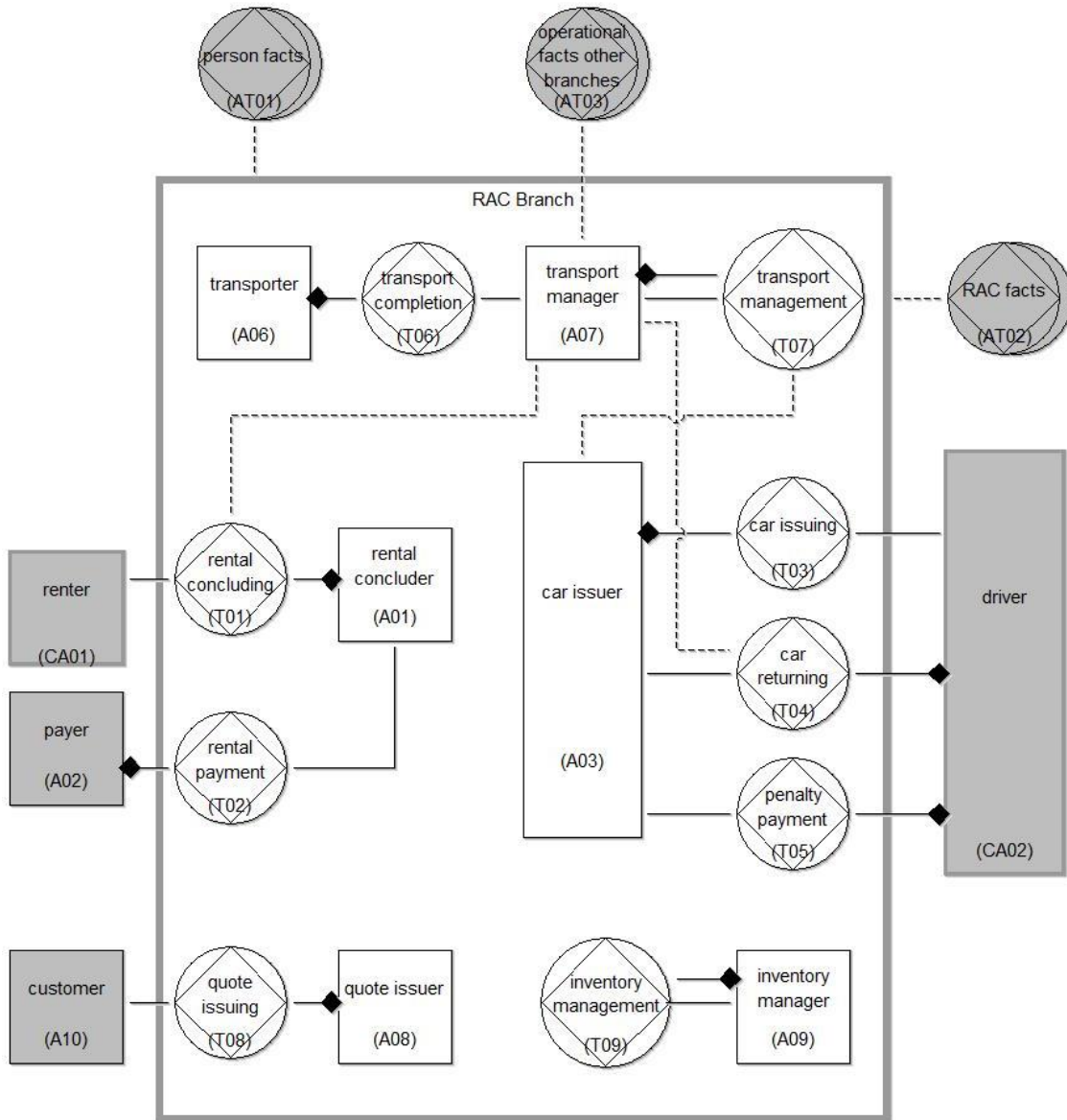


Figure 41: OCD for Rent-a-Car (Perinforma, 2017)

transaction ID	transaction kind	product ID	product kind
(T01)	rental concluding	(P01)	Rental is concluded
(T02)	rental payment	(P02)	the rent of Rental is paid
(T03)	car issuing	(P03)	the car of Rental is issued
(T04)	car returning	(P04)	the car of Rental is returned
(T05)	penalty payment	(P05)	the penalty of Rental is paid
(T06)	transport completion	(P06)	Transport is completed
(T07)	transport management	(P07)	transport management for Day is done
(T08)	quote issuing	(P08)	Quote is issued
(T09)	inventory management	(P09)	inventory management for Month is done

Figure 42: TPT for Rent-a-Car

As seen in Figure 41, the TKs ‘quote issuing’ and ‘inventory management’, with their respective initiating and executing ARs, have been added to the OCD described in Perinforma (2017). These two additions are also reflected in the TPT, as seen in Figure 42.

1.26 OCD to BPMN transformations for Rent-a-Car

The four scenarios are represented in the OCD by the following TKs:

- *Scenario 1:* quote issuing – Here the user-selected TK is initiated by an AR outside the SOI and executed by an AR within the SOI. The AR does not initiate another TK.
- *Scenario 2:* rental payment – The user-selected TK forms part of another TK (i.e., rental concluding).
- *Scenario 3:* inventory management – The user-selected TK is self-initiating. Also, the AR (module reviser) does not initiate another TK.
- *Scenario 4:* transport management – This TK plays the role of a parent for a part TK (i.e., transport completion).

1.26.1 Scenario 1 transformation for Rent-a-Car

The transformation for the TK ‘quote issuing’ is done in Figure 43. From the figure it can be seen that the descriptions of the model elements automatically cater for the situations. This is different from those in Figure 33.

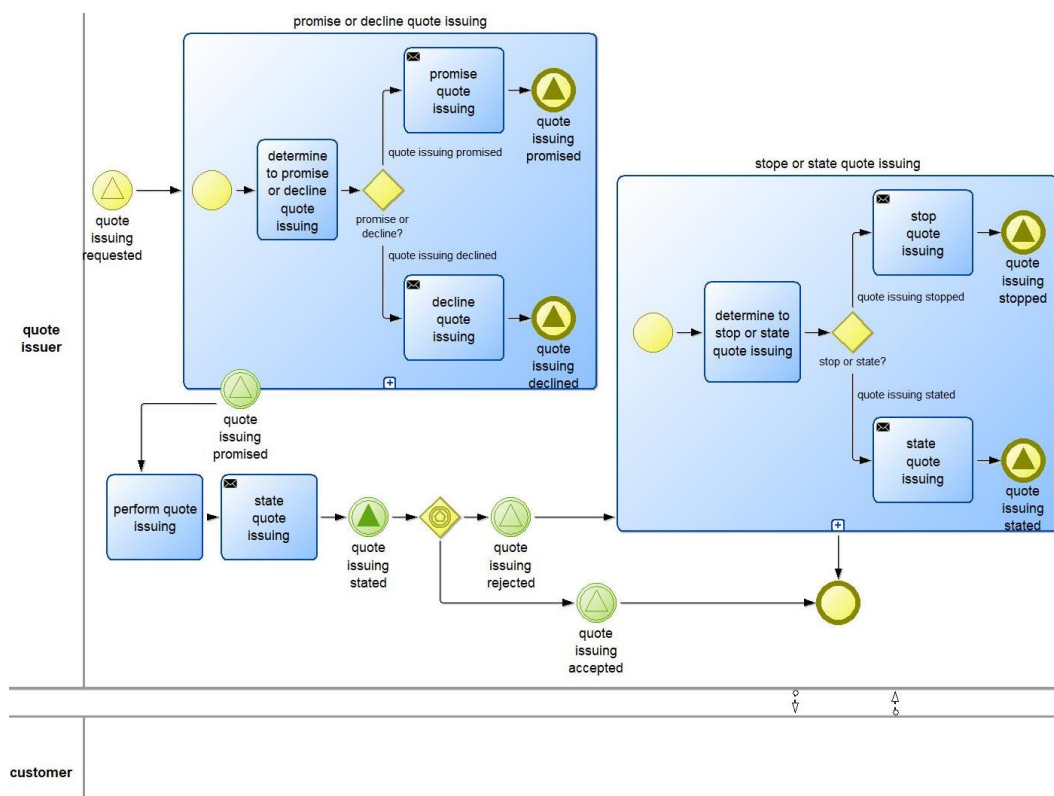


Figure 43: Scenario 1 transformation Rent-a-Car

1.26.2 Scenario 2 transformation for Rent-a-Car

For the second scenario, the transformation becomes problematic. Figure 44 below shows the PSD for the transaction taking place in Scenario 2 (Perinforma, 2017). The prerequisite fact from the parent TK (T1) is *requested* for the act of the part TK (T2) *request*. As per the PSD however, for the reverse relationship, the part TK has two prerequisite facts ‘*promised*’ and ‘*accepted*’. Furthermore, while DMT caters for the four acts previously specified (*request*, *promise*, *state* and *accept*), the PSD below refers to an *execute* act. Neither DMT or the specifications by De Vries and Bork (In review) makes allowance for a situation such as this.

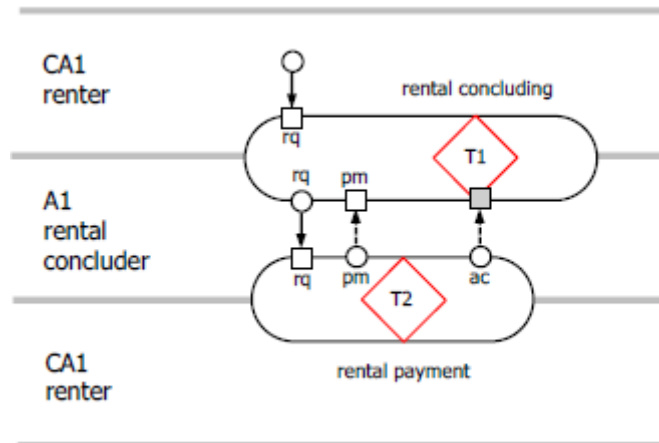


Figure 44: PSD of Scenario 2 (Perinforma, 2017)

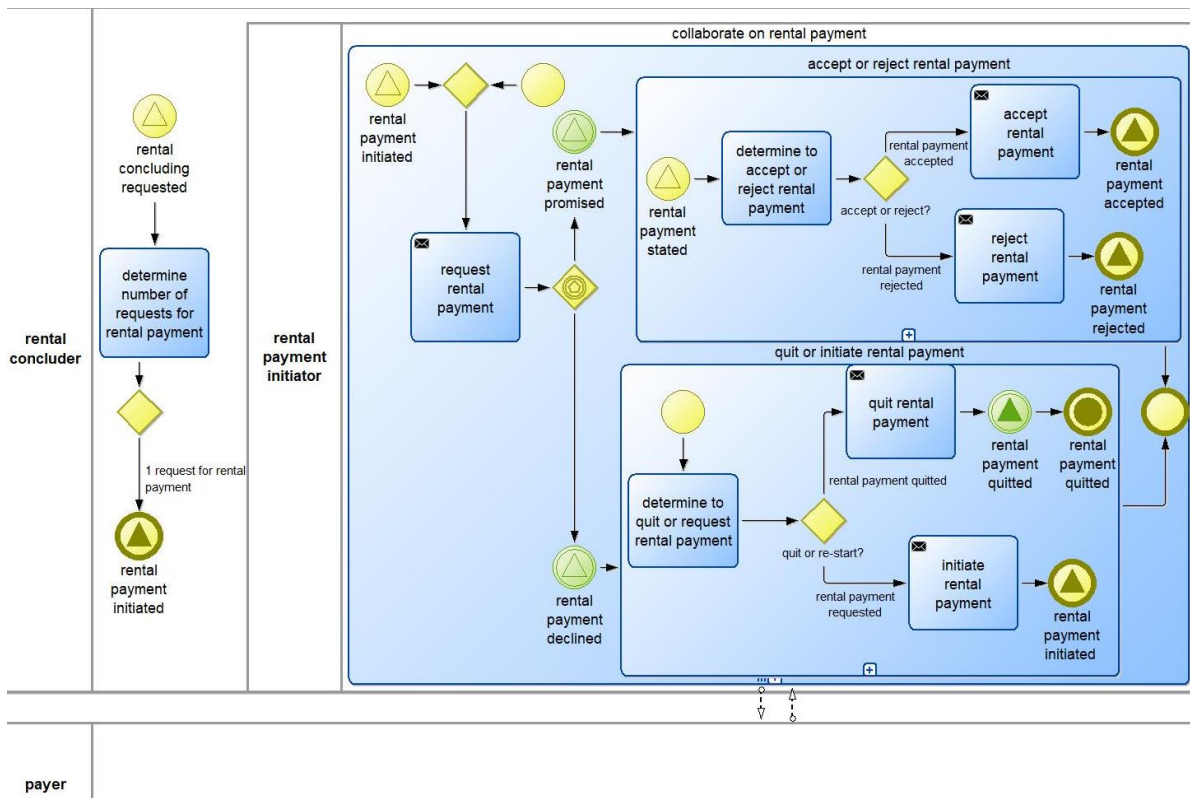


Figure 45: Scenario 2 transformation Rent-a-Car

For reference, the incomplete BPMN diagram generated by DMT is shown in Figure 45 above. Another instance that DMT does not cater for can be seen to be the gateway that exists after the task ‘determine number of requests for rental payment’. This gateway is not necessary as there is no split in the sequence flow as in the college case.

From this attempted transformation it can be seen that the specifications by De Vries and Bork (In review) used for this initial version of DMT are incomplete.

1.26.3 Scenario 3 transformation for Rent-a-Car

The third scenario, given by the TK ‘inventory management’, requires no user input, and so can be generated by the user simply by selecting the relevant TK. The resulting BPMN diagram can be seen below in Figure 46.

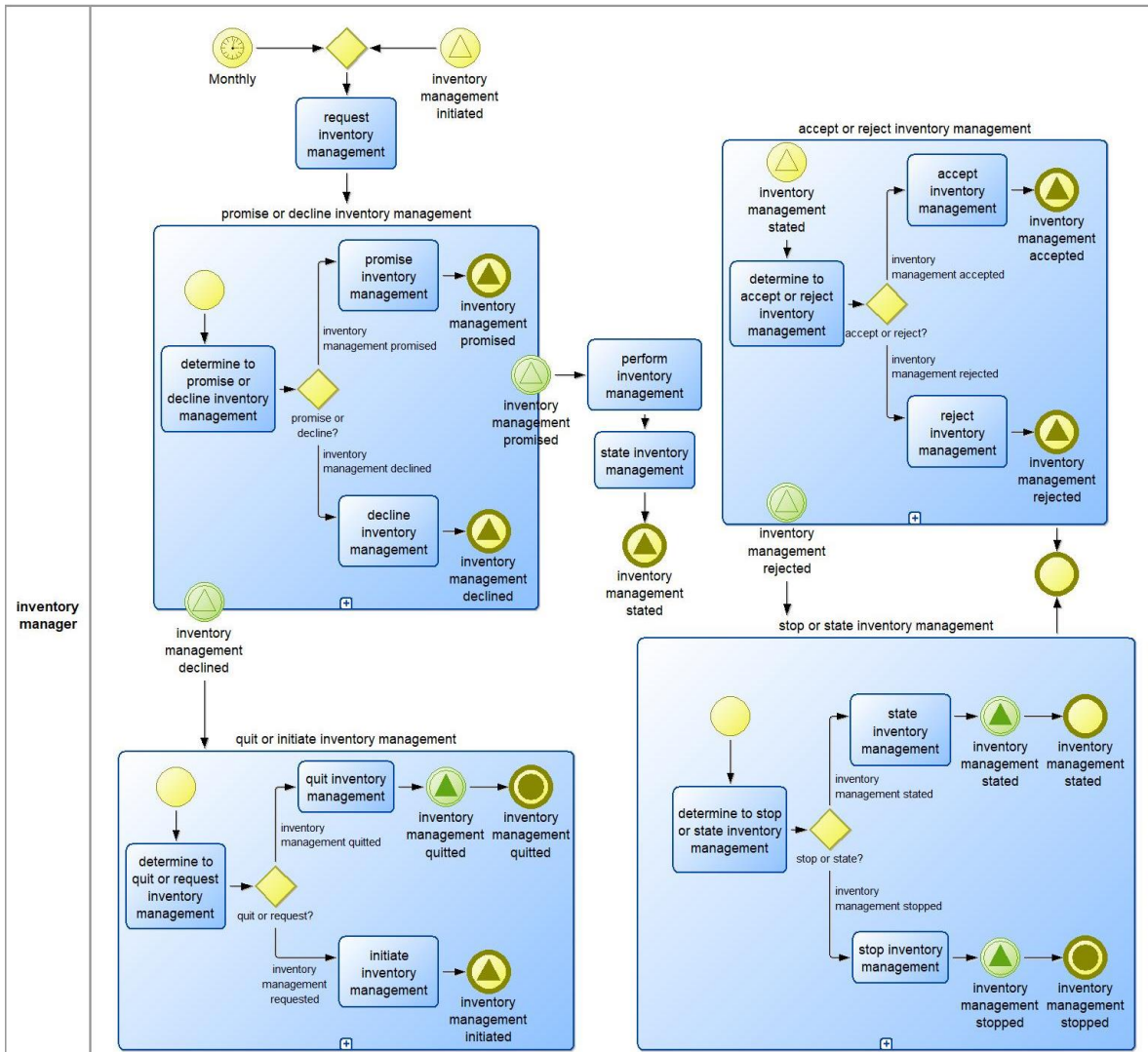


Figure 46: Scenario 3 transformation Rent-a-Car

1.26.4 Scenario 4 transformation for Rent-a-Car

For the fourth scenario, the parent-to-part structure is shown in Figure 47 and the generated BPMN diagram in Figure 48. As can be seen in the BPMN diagram, a gateway was generated to connect the tasks: ‘state transport management’ and ‘determine number of requests for transport management’. For the case of Rent-a-Car this gateway should not exist as it is only connected to one output. This is a flaw with the current version of DMT which only occurs once a transformation for Scenario 4 is done with only one part TK.

from parent	prereq parent fact	act of part	# of part instances	to part	prereq part fact	act of parent	# of prereq part instances
transport management	promised	request	0..*	transport completion	accepted	accept	0..*

Figure 47: Parent part structure scenario 4

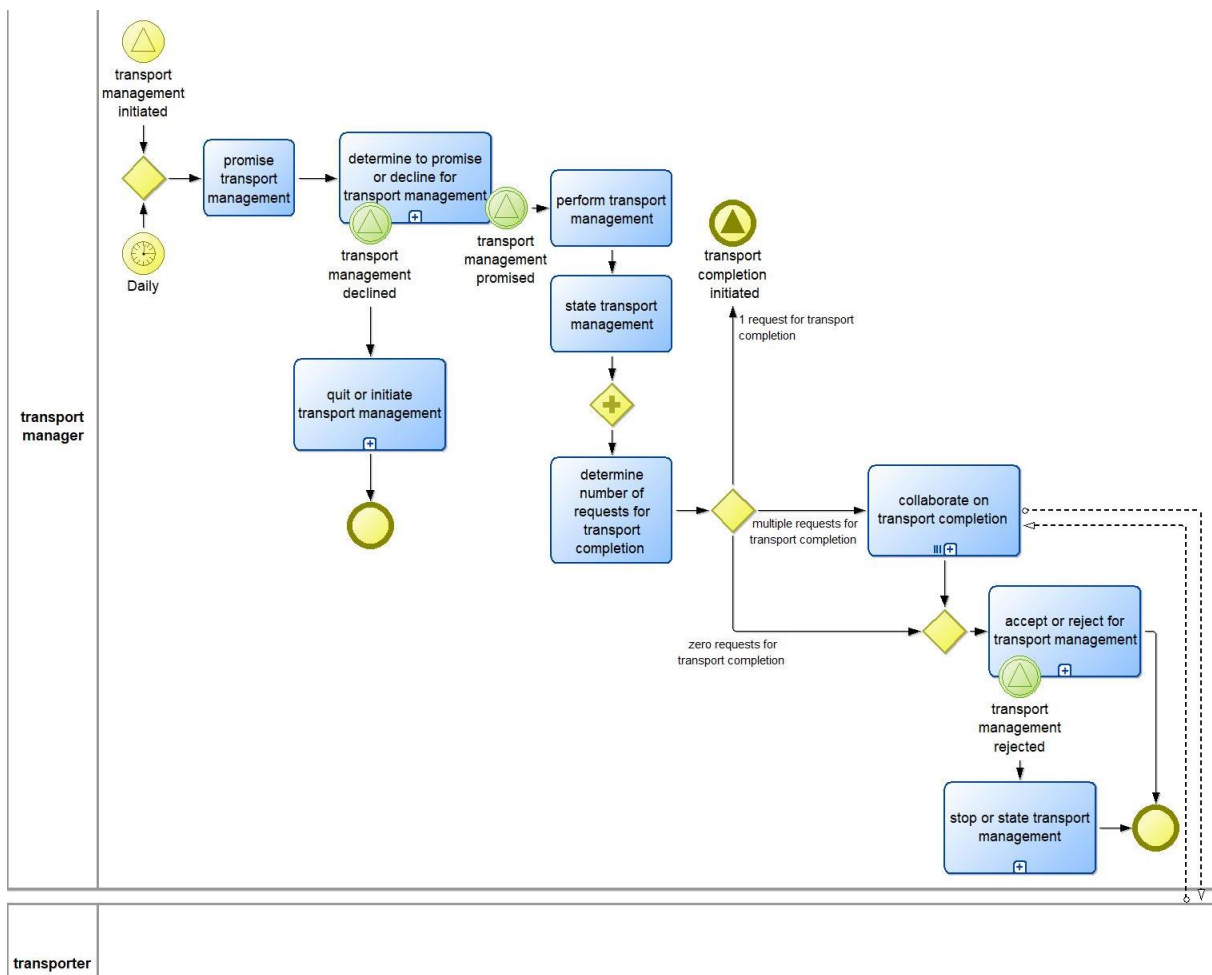


Figure 48: Scenario 4 transformation Rent-a-Car

The goal of this chapter has been to demonstrate the use of DMT by first modelling the OCD and TPT diagrams of two case studies, and then using the tool to perform transformations from the OCD to BPMN collaboration diagrams. This has been done although the second case highlighted some pitfalls with both DMT and the transformation specifications where the main area of concern is the user input. In the next chapter the usability of the tool is discussed.

Chapter 7

DMT evaluation

DMT's evaluation consists of two parts: utility and usability. Chapter 6 has already demonstrated the utility of DMT, demonstrating its main feature: the semi-automatic transformation of a user-selected TK on the OCD to a BPMN collaboration diagram. This chapter provides additional evaluation results for the *utility* of DMT, complying with the DEMO meta-model presented in DEMOSL 3.7 (with extensions). The meta-model validation was done by composing several tests to measure compliance.

In addition, the *usability* of DMT was tested by a group of post-graduate participants, who had to use the college case to compile an OCD and TPT. Afterwards they had to complete a SUMI questionnaire. The participants also made suggestions about potential problem areas. How these were addressed for the newest version of DMT is also described below.

1.27 Meta-model compliance

The meta-model adherence results for the compliance test are displayed in Table 12 below. These are in accordance with the previously discussed DEMOSL 3.7 meta-model with extensions. This meta-model is repeated in Figure 49 below for reference purposes.

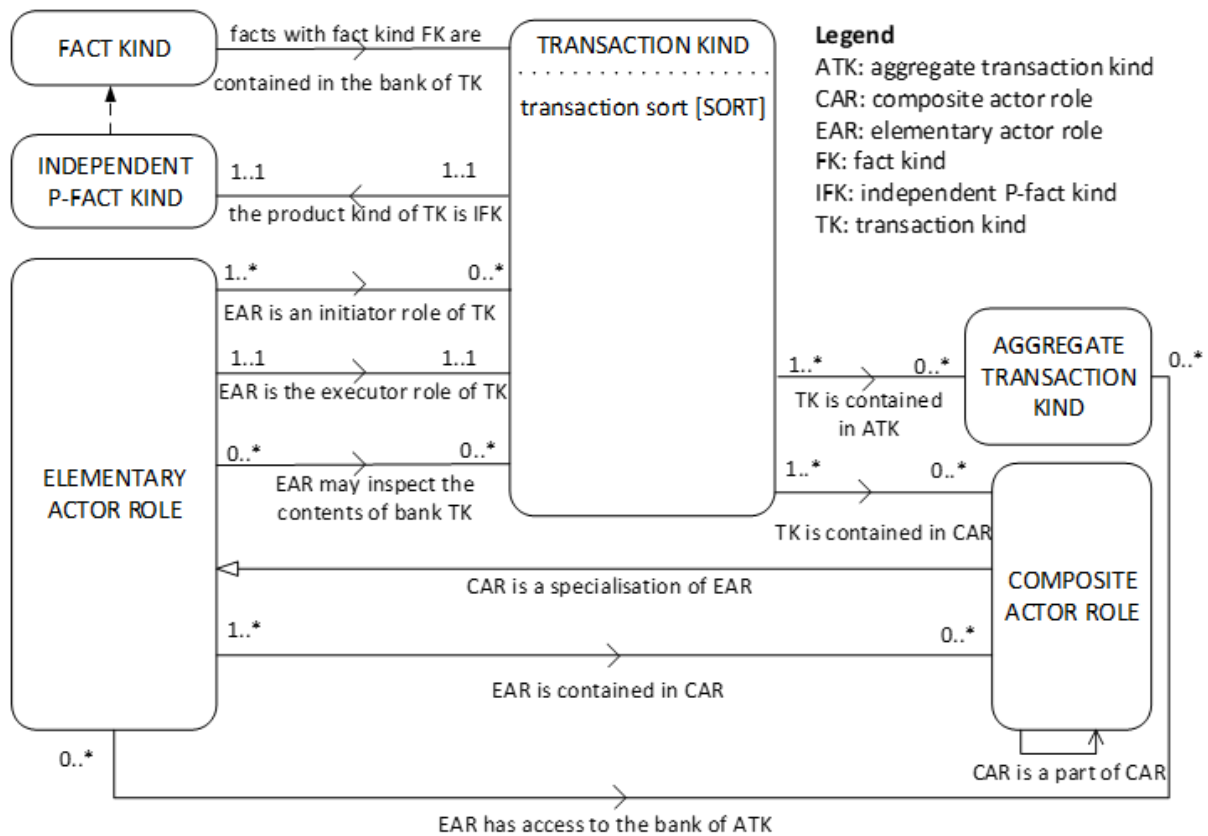


Figure 49: CM meta-model

Table 12 below shows what existence rules were tested, their test cases, and the test results. Where DMT does not explicitly enforce a cardinality, the test result provides more information.

Table 12: Meta-model tests

Existence rule	Test case and expected results	Test results
<p><i>From TK:</i></p> <p>1..1 the product kind of TK is IFK 1..1</p>	<p>Forward: When T01 is defined, it should be linked to one product kind. When viewing the TPT, the single product kind should be displayed for the TK.</p> <p>Reverse: When defining the product kind P02, it should be for one TK.</p>	<p>Forward: The Model Analysis/Validation feature highlights production kinds with no product kind defined. Thus mandatory 1..1 is not enforced.</p> <p>Reverse: The product kind ID is automatically generated and is not editable. The system also blocks any attempt to duplicate product kind names.</p>
<p>1..* TK is contained in ATK 0..*</p>	<p>Forward: <i>Outside Sol:</i> For DEMOSL 3.7 it should not be possible to indicate the parent-part relationship. <i>Inside Sol:</i> It should be possible to create T01 in accordance with Figure 31, where T01 is not contained in an ATK. It should be possible to create T02, where T02 is contained in the ATK T05. T05 is an ATK, since it has multiple parts (i.e., T02, T03, T06, T07).</p> <p>Reverse: <i>Outside Sol:</i> For DEMOSL 3.7 it should not be possible to indicate the</p>	<p>Forward: Behaviour is in accordance with the test case description.</p> <p>Reverse: An ATK placed within the SOI is coloured red to indicate non-validity. The Model Analysis/Validation feature also highlights it as an error.</p>

Existence rule	Test case and expected results	Test results
	parent-part relationship. <i>Inside Sol</i> : It should not be possible initially to model T05 as an ATK, grey-shading T05, without indicating the parts – i.e., without modelling the parts T02, T03, T06, T07). The software should only allow T05 as a composite (without parts) if created outside the Sol.	
1..* TK is contained in CAR 0..*	Forward: It should be possible to create T01 that is not contained within the CAR CA00 (College). It should also be possible to create T06 as a part within CAR CA00 (College). Reverse: It should be possible to create CAR CA00 (College) with multiple TKs – namely, T04, T05 and T06.	Forward: Behaviour is in accordance with the test case description. Reverse: Behaviour is in accordance with the test case description.
<i>From EAR:</i>		
1..* EAR is an initiator role of TK 0..*	Forward: It should be possible to create A01 as per Figure 31, where A01 is not initiating other TKs. It should be possible to create A05, initiating T02, T03 and T06. Reverse: It should be possible to create T06, initiated by A05 as in Figure 31. It should also be possible to create T06, also initiated by A01 (i.e., A01 is also initiating T06), even though this scenario is not evident on Figure 31.	Forward: Behaviour is in accordance with the test case description. Reverse: Behaviour is in accordance with the test case description.
1..1 EAR is the executor role of TK 1..1	Forward: When A01 is modelled as an elementary actor role without any execution link attached, a validation message should be shown when validating the model. When A01 is modelled as an elementary actor role, it should not be possible to connect both A01 as the executor for T01, and A01 as the executor for another TK – say, T08 (supervision). Note that T08 is not displayed on Figure 31, but has to be created temporarily to do this test. Reverse: When T01 is modelled as a TK without any execution link attached, a validation message should be shown when validating the model. When T01 is linked via an execution link to A01 and T01 is linked to another EAR – say, A08 (supervisor) – an error message should appear to indicate that a TK may only have one executor.	Forward: Behaviour is in accordance with the test case description. Reverse: Behaviour is in accordance with the test case description. A warning message is displayed, and the second connection is not possible.
0..* EAR may inspect the contents of bank TK 0..*	Forward: When A01 is created without any inspection links, no validation rules should appear when saving the model. It should also be possible to link A01 to T02 via an inspection link, as well as A01 to T06, with no validation errors appearing when the model is saved. Reverse: It should be possible to create the TK T03 with no inspection links attached. It should also be possible to	Forward: Behaviour is in accordance with the test case description. Reverse: Behaviour is in accordance with the test case description.

Existence rule	Test case and expected results	Test results
	create an inspection link between T02 and A01, as well as an inspection link from T02 to A06.	
0..* EAR is contained in CAR 0..*	<p>Forward: It should be possible to create an actor role – say, A08 (supervisor) – within the SoI, but outside the CAR CA00 (College), without displaying validation errors when saving the model. It should be possible to create A06 as an EAR embedded in the CAR CA00 (College), also embedded in the CAR CA01 (Controller).</p> <p>Reverse: It should be possible to create the CAR CA00 (College) without embedding any TKs, linking it via an execution link to T01. It should be possible to create the CAR CA00 (College) with multiple embedded EARs – i.e., A01, A04, A05 and A06.</p>	<p>Forward: Behaviour is in accordance with the test case description.</p> <p>Reverse: Behaviour is in accordance with the test case description.</p>
0..* EAR has access to the bank of ATK 0..*	<p>Forward: It should be possible to create the EAR A01 with no inspection links to AT01 and AT02 – i.e., no validation errors on saving. It should also be possible to create EAR A04 with inspection links to AT03 and AT04 without validation errors on saving.</p> <p>Reverse: It should be possible to create AT5 without any inspection links attached, with no validation errors on saving. It should be possible to create both an inspection link from AT04 to A04 and an inspection link from AT04 to A06, without validation errors on saving.</p>	<p>Forward: Behaviour is in accordance with the test case description.</p> <p>Reverse: Behaviour is in accordance with the test case description.</p>
<i>From CAR:</i>		
CAR is a specialisation of EAR	The relations of the EAR should also be available for the CAR (Mulder, 2019a). Thus, when creating CAR CA00 (College) without any embedded detail, it should be possible to link the CA00 via an execution link to T01, link CA00 via an initiation link to T07, link CA00 via an initiation link to T02, link CA00 via an initiation link to T03, and link CA00 via an inspection link to AT04.	Behaviour is in accordance with the test case description.
CAR is a part of CAR	As explained by Mulder (2019a), the SoI is a special case of a CAR. It should be possible to create a CAR – e.g., CA00 (College) – within the SoI without any validation errors when saving the model.	Behaviour is in accordance with the test case description.

As seen in the table above, DMT adheres to the DEMO meta-model specifications as interpreted by the author of this dissertation.

1.28 SUMI survey and results

To evaluate the usability of DMT, the participants had to recreate the OCD and subsequent TPT displayed in Figure 31 and Figure 32. They were also required to perform the transformation

for Scenario 2 in line with Figure 35. Figures of both the OCD and the TPT were supplied, as well as the exact parent-to-part relationship specifications for a Scenario 2 transformation.

The SUMI questionnaire was completed by 34 people. A summary of the results is displayed below in Table 13 and Figure 50. The SUMI test consists of five sub-categories (efficiency, affect, helpfulness, controllability, and learnability), which are consolidated into a global scale (Sauro & Lewis, 2012).

The results gathered for DMT are thus very positive, with a global mean score of 56.50. From Table 13, it can be seen that the highest rated subcategory was *affect* with a score of 60.85. *Affect* measures whether a participant has an overall pleasant emotional feeling of being mentally stimulated. A positive score for *affect* thus means that the participants did not feel stressed or frustrated while completing the task at hand.

Table 13: SUMI results

	<i>Mean</i>	<i>Standard dev.</i>	<i>Median</i>
<i>Global</i>	56.50	10.86	56.50
<i>Efficiency</i>	54.03	13.53	54.50
<i>Affect</i>	60.85	10.67	64.50
<i>Helpfulness</i>	54.97	10.94	55.00
<i>Controllability</i>	55.47	10.33	54.50
<i>Learnability</i>	54.65	11.41	57.00

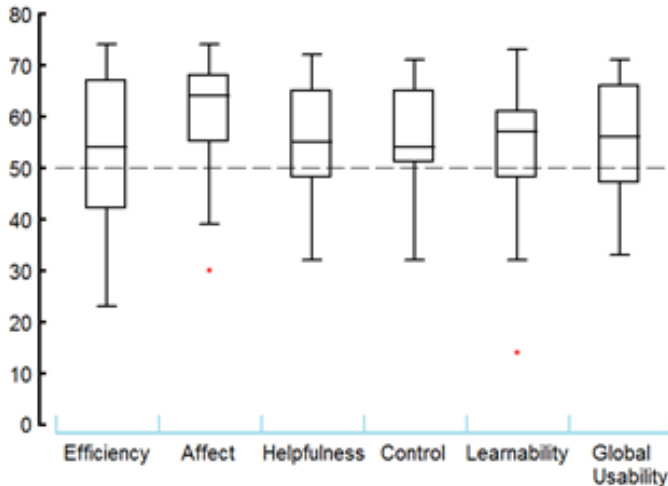


Figure 50: SUMI results

Furthermore, none of the other sub-categories scored below 50. This indicates that the participants found DMT to be generally user-friendly. More information on the SUMI

questionnaire, such as how the category scores are calculated, can be found on the SUMI website (<http://sumi.uxp.ie/>).

1.29 *Individual participants' feedback*

Aside from the overall results above, SUMI also provides an opportunity for individual qualitative feedback. This, as well as feedback given to the author while he facilitated the practical modelling session, is discussed below.

The feedback indicated that 31 out of the 34 participants regarded DMT as *important* or *extremely important* to the task they were required to do. Furthermore, the participants indicated both what they liked and how they felt that DMT could be improved, summarised in Table 14 below.

The feedback displayed in Table 13 is split into two categories: *positives* and *areas for improvement*. For both of these areas, the table is divided into the following columns: Category, Number of participants, and Description. The *Category* column divides the feedback into general topics of interest pertaining to either positive aspects or areas for improvement. The *Number of participants* column lists the number of participants who commented on a topic. The final column provides a description of the category being discussed.

Table 14: Participant feedback

<i>Positives</i>		
Category	Number of participants	Description
<i>Ease of use</i>	15	How easy it is to use DMT overall.
<i>Intuitiveness</i>	7	Refers to the level at which the user instinctively knew how to use DMT functionality.
<i>Model transformation</i>	7	Sees the DEMO to BPMN transformation as useful and a possible asset.
<i>Interface</i>	6	The user saw the interface as attractive and intuitive in its design.
<i>Drag-and-drop</i>	4	The drag-and-drop functionality of the tool.
<i>Areas for improvement</i>		
<i>Link usage</i>	7	Refers to difficulty in connecting relational links between relevant objects.
<i>General usage</i>	6	Refers to general usage that could be improved.
<i>Error handling</i>	4	Refers to error messages not being clear enough, or to the potential for help screens that explain what to do.
<i>Menu</i>	3	Refers to the layout of the menu in DMT being unclear.

1.30 *Improvements from feedback*

Although the positive feedback provides some indication of success with DMT, it was the negative feedback that proved to be more useful to the author. The feedback was examined, and either used to make improvements to the tool or added to the list of future improvements.

The table below illustrates some of the concerns that have already been addressed. It contains the category of improvement, the specific area that was lacking, and the fix that was applied.

Table 15: Feedback and improvements

Category	Feedback	Fix
<i>General usage</i>	When generating the TPT, the TK ID and the product kind ID should automatically match without the user having to specify both.	The production kind ID is now no longer editable by the user, and automatically conforms to the TK ID.
<i>General usage</i>	The user may wish to change details in the OCD, such as font sizes and name positions.	The user can now change the font size of both element names and IDs, and choose what information to display in the diagrams and where to display this information.
<i>Error handling</i>	It is unclear what needs to be done to fix a problem from the error messages displayed to the user.	Error messages have been reprogrammed to include clear information on what the user needs to do to fix an error.

Regarding future improvements, the author of this dissertation observed that comments relating to areas of improvement referred to functionality of DMT that wasn't difficult but was rather unclear. Thus, a vision for the future of DMT is to provide clear information to the user about the use of the tool. This will, first of all, include pop-ups and help screens within DMT to explain different functionalities. Then tutorial videos, example models, and detailed 'read-me' documentation will be created to ensure that new DMT users have everything they need to use the tool without difficulty.

Chapter 8

Conclusion and future work

The goal of this dissertation was to create a modelling tool that could create the DEMO's CM (i.e., OCD and TPT), that was based on the most recent specification language (at the time it was created), and that could facilitate model transformations. It was also stated that this tool should be user-friendly and available free of charge to educational institutions. To achieve this goal a number of research questions were asked. Section 8.1 summarises the results obtained on every research question. Section 8.2 provides ideas for future research and section 8.3 presents a final conclusion.

1.31 Summary per research question

Question 1: What are the available construction modelling tools? (Answered in Section 1.18)

A number of CM tools were found, as identified by Mulder (n.d.) and the University of Pretoria.

Question 2: Do these tools meet the University of Pretoria's (main stakeholder's) requirements – i.e., are they free-to-use and easy-to-use, do they enable vertical consistency with other DEMO models, and enable horizontal transformation to other model types? (Answered in Section 1.18)

It was concluded that none of the investigated tools met the University of Pretoria's requirements. None of the tools were able to model all four of the DEMO aspect models; none were based on a sound DEMO meta-model (vertical consistency); and none could facilitate horizontal transformations. Although DMT as created in this dissertation can only model the CM of the aspect models, it is based on a sound meta-model and enables horizontal transformations.

Question 3: Does the problem instance feature as a class-of-problems in the existing literature? (Answered in Section 1.9.2)

It was found that the problem instance does feature as a class-of-problems in the existing literature. An SLR by Awadid and Nurcan (2019) revealed that both horizontal and vertical inconsistencies exist in the field of conceptual modelling.

Question 4: What are the key concepts used to describe modelling constructs and the translation or transformation of constructs? (Answered in Section 1.6)

Table 1 provides a comprehensive list of the concepts used. This was gathered throughout this dissertation from the relevant literature.

Question 5: What are the vertical inconsistencies between different DEMO models, and how can they be avoided? (Answered in Sections 1.9.4 and 1.9.5).

From the SLR conducted, only one source – Mulder (2019b) – identified the meta-model as a potential source of inconsistencies between different DEMO models. From that article it can be gathered that the inconsistencies within the meta-model could prove troublesome when automatic or semi-automatic transformations between the aspect models are considered.

The same author also proposed that the DEMOSL 3.7 meta-model be extended to be more consistent (Mulder, 2019a). Given that DEMOSL 3.7 was the newest version of the DEMO meta-model, and had the greatest potential for future versions, DEMOSL 3.7 with Mulder's extension was chosen as the base meta-model for DMT.

Question 6: What approaches have been used to accommodate multi-view modelling up to now? (Answered in Section 1.9.3)

The SLR by Cicchetti, Ciccozzi, and Pierantonio (2019a) revealed that, so far, few multi-view modelling approaches have incorporated model transformations. The SLR also showed that, although the majority of approaches do take steps to address inconsistencies in multi-view modelling, it is still an area of concern for modelling tools.

From the SLR, the importance of tool support was highlighted the most within the literature. This indicated that the modelling community expects not only a well performing tool, but also background support.

Finally, the SLR found that a common problem in the approaches investigated is the lack of expressiveness in describing multi-view artefacts and the way they react. A need was identified for more general functionality of modelling tools aside from just creating base models.

Question 7: Have other authors attempted to transform DEMO models into other model types, and, if so, how was this done? (Answered in Sections 1.9.4, 1.9.5 and 1.12)

A number of literature sources were identified that attempted to transform DEMO models into other model types (ArchiMate, Petri Net, XML, and BPMN). The majority of the sources attempted to transform DEMO into other model types by mapping corresponding concepts between the meta-models. None of the sources implemented the transformations in a modelling tool, but rather provided the theoretical techniques for the transformation.

Given the context of this dissertation, the literature on DEMO-to-BPMN was discussed more thoroughly. Of the techniques discussed, it was concluded that De Vries and Bork (In review) provide the most up-to-date and all-encompassing transformation specifications. Thus, their specifications for OCD to BPMN were incorporated into DMT.

Question 8: How should the new modelling tool be developed? (Answered in Chapter 5)

The tool was developed to adhere to the requirements described in Section 1.20. The development consisted of two parts: (1) The development of the CM within the ADOxx platform by incorporating the CM elements in the existing BPMN library; and (2) the background coding to facilitate the transformations.

Question 9: How should the new modelling tool be evaluated? (Answered in Chapter 7)

DMT's evaluation was divided into two sections: (1) meta-model compliance, and (2) usability. The meta-model compliance was tested by constructing a set of tests to ascertain whether DMT adheres to the prescribed meta-model. The usability was tested by having a group of post-graduate participants model a case study within DMT and then fill out the SUMI questionnaire, which is commonly used to measure the usability of a tool (Sauro & Lewis, 2012).

DMT was found both to adhere to the prescribed meta-model (see Table 12) and to be user-friendly (see Table 13). Furthermore, some of the deficiencies in DMT identified by the participants have already been addressed (see Section 1.30).

Question 10: What are the constructional components of the tool? (Answered in Chapter 5)

The constructional components of the tool are (1) the modelling platform for modelling the OCD and TPT representations of the CM; then (2) the transformation capability from a created OCD, where the user selects a particular transaction kind for transformation, to the BPMN collaboration diagram.

Question 11: What case demonstrations provide sufficient evidence to demonstrate the utility of the tool? (Answered in Chapter 6)

Two cases were used to demonstrate the utility of the tool: the processes at the fictitious college used by De Vries and Bork (In review), and the Rent-a-Car case (with some additions) found in Perinforma (2017). These cases were chosen because they not only showcase the different OCD elements, but also include the four scenarios identified by De Vries and Bork (In review).

The Rent-a-Car case proved especially useful as it highlighted some areas where the transformation specifications were incomplete. DMT can thus be improved by facilitating these changes to the transformation specifications.

Question 12: How useful is the tool in terms of its expected utility? (Answered in Chapter 7)

For both the college and the Rent-a-Car cases, the relevant OCD and TPT diagrams were first created. Next, the transformations from the OCD to the BPMN collaboration diagram were performed for all four scenarios.

Thus enough evidence was provided that DMT can both create the relevant CM diagrams and facilitate semi-automatic transformations from the OCD to BPMN collaboration diagrams. It can therefore be concluded that the DMT tool is useful in respect of its expected utility.

1.32 Future work

The next step in DMT's development is to expand it to include the newest version of the DEMO specification language (i.e., DEMOSL 4.5). This will include not only the implementation of the newest version of the CM, but also the other three aspect models.

Although the transformation specifications by De Vries and Bork (In review) was the most complete at the date of the SLR, there were some transformation sub-scenarios that it did not cater for. Furthermore, given that the specifications are based on an old version of the DEMO specification language, it needs to be updated, according to the latest DEMOSL 4.5. Once this is done, DMT can be updated to facilitate the new specifications.

Given the recent movement towards process automation, it is relevant to create BPMN models that enable simulation. The specifications discussed in this dissertation only aimed to ensure an accurate transformation from DEMO to BPMN without considering whether the created BPMN models would allow for simulation. The transformed BPMN models need to be further adapted to facilitate simulations.

DMT has already been demonstrated at a conference earlier this year, with positive feedback from conference attendees (see (Gray et al., 2020)). The implementation and testing of DMT in a professional environment would also be beneficial for its future development, by getting feedback from individuals who – unlike the participant group used in this dissertation – would attempt to apply DMT within a real-world enterprise context.

1.33 Concluding remarks

It was concluded that the study provides sufficient evidence to support the thesis statement, i.e.: “A new DEMO tool, developed on the ADOxx platform, and based on a sound meta-model that meets the requirements (including a *free-to-use* and *easy-to-use* tool for academics and students that *enables vertical consistency between DEMO models*, and that *enables horizontal transformation to other model types*, such as BPMN models) will help the University of Pretoria as well as the wider enterprise engineering community to model enterprises within a multi-view modelling paradigm.”

Chapter 9

References

- ADOxx (n.d). "Inheritance/Dependencies of ADOxx Dynamic Metamodel." Retrieved 22 April, 2020, from <https://www.adoxx.org/live/predefined-abstract-classes-dynamic->.
- Albani, A., & Dietz, J. L. (2006). *The benefit of enterprise ontology in identifying business components*. Paper presented at the IFIP World Computer Congress, TC 8.
- Awadid, A., & Nurcan, S. (2019). Consistency requirements in business process modeling: A thorough overview. *Software & Systems Modeling*, 18(2), 1097-1115. doi:10.1007/s10270-017-0629-2
- Badal, V. (2009). *A roadmap for the transition from paper to digital records*. Retrieved 15 November, 2019, from <http://resolver.tudelft.nl/uuid:8add2640-47a1-4a59-9184-35e22837ac89>
- Bakhshandeh, M. (2016). *Ontology-driven analysis of enterprise architecture models*. PhD thesis, Universidade De Lisboa.
- Boedhram, V. R. K., & Algoe, S. S. R. W. (2009). *DEMO applied to financial services*. Retrieved 15 November, 2019, from <http://resolver.tudelft.nl/uuid:1999a3fb-36c2-4898-95a4-bc2040858780>
- Bork, D. (2016). *A development method for the conceptual design of multi-view modeling tools with an emphasis on consistency requirements*. Bamberg: WorldCat database.
- Bork, D., Buchmann, R., & Karagiannis, D. (2015). Preserving multi-view consistency in diagrammatic knowledge representation modeling and simulation-based solutions. In *Climate change and energy dynamics in the Middle East: Modeling and simulation-based solutions* (pp. 177-182). Cham: Springer International Publishing.
- Bork, D., & Karagiannis, D. (2014). *Model-driven development of multi-view modelling tools: The MuViewMoT approach*. To appear in: Holzinger, Cardoso, Cordeiro, van Sinderen and Mellor (eds.), Proceedings of the 9th International Joint Conference on Software Paradigm Trends (ICSOFT-PT) 2014, pp. IS-11-IS-23, SCITEPRESS Digital Library, Vienna, Austria
- Boulding, K. E. (1956). General systems theory: The skeleton of science. *Management Science*, 2(3), 197-208. doi:10.1287/mnsc.2.3.197
- Cicchetti, A., Ciccozzi, F., & Pierantonio, A. (2019). Multi-view approaches for software and system modelling: A systematic literature review. *Software and Systems Modeling*, 18(6), 3207-3233. doi:10.1007/s10270-018-00713-w
- De Kinderen, S., Gaaloul, K., & Proper, H. A. (2014). Bridging value modelling to ArchiMate via transaction modelling. *Software & Systems Modeling*, 13(3), 1043-1057.
- De Vries, M., & Bork, D. (In review). *Transforming organisation design knowledge into executable business processes: An approach based on DEMO and BPMN* (Unpublished).
- De Vries, M., van der Merwe, A., & Gerber, A. (2017). Extending the enterprise evolution contextualisation model. *Enterprise Information Systems*, 11(6), 787-827. doi:10.1080/17517575.2015.1090629

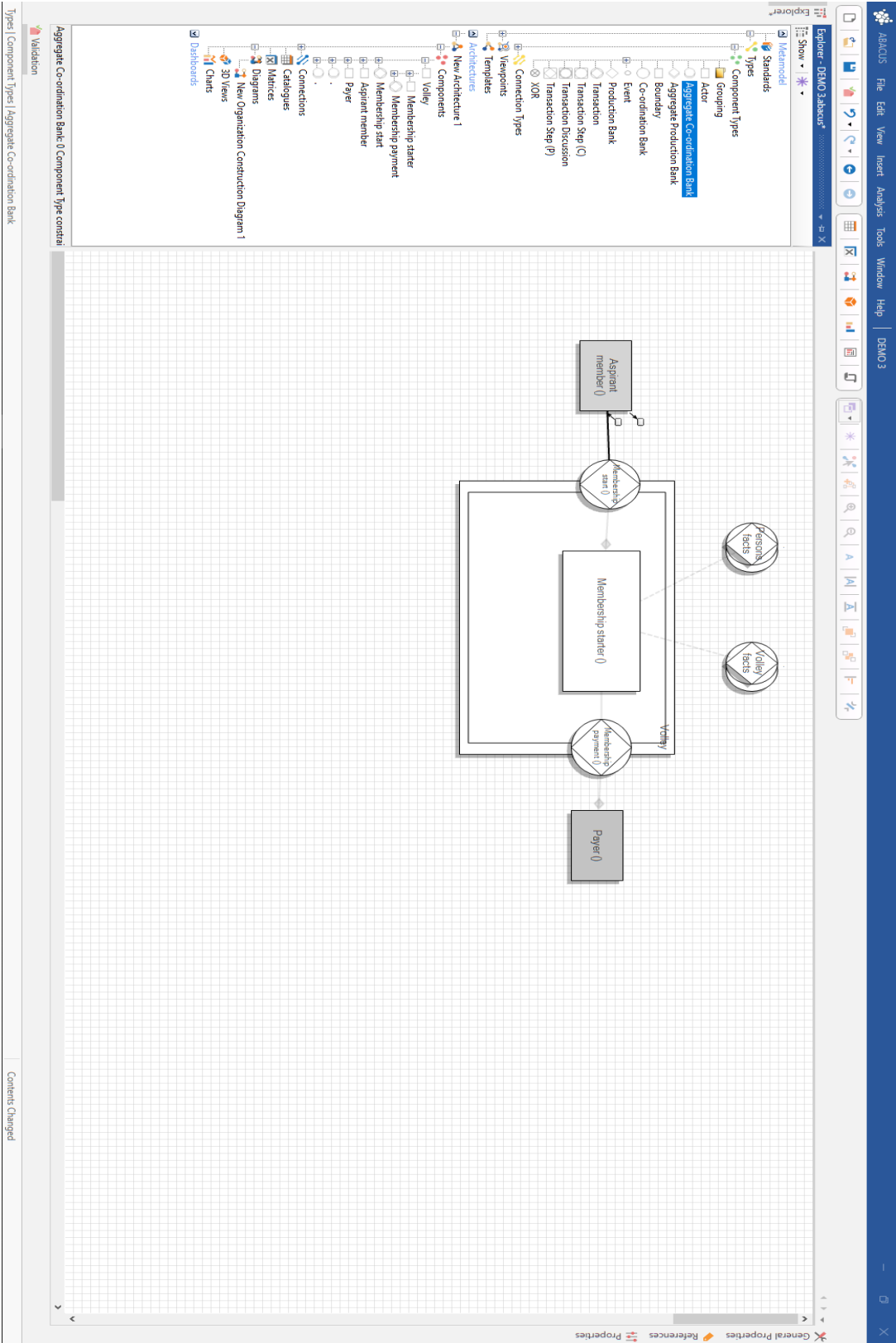
- Decosse, C., Molnar, W. A., & Proper, H. A. (2014). *What does DEMO do? A qualitative analysis about DEMO in practice*. Paper presented at the Advances in Enterprise Engineering VIII, Portugal.
- Dietz, Jan L. G. and Hans B. F. Mulder. 2020. *Enterprise Ontology: A Human-Centric Approach to Understanding the Essence of Organisation*. Cham: Springer. Retrieved August 6, 2020, from <https://ebookcentral-proquest-com.uplib.idm.oclc.org/lib/pretoria-ebooks/detail.action?docID=6181555>.
- Dietz, J. L. G., & Mulder, M. A. T. (2017). *DEMOSL – 3: DEMO specification language*. Retrieved November 13, 2019, from <https://demo.nl/mdocs-posts/demo-specification-language-3-7/>
- Ding, H. (2016). *Integrating value modeling into ArchiMate*. University of Twente, Netherlands.
- Dumas, M., La Rosa, M., & Mendling, J. (2018). *Fundamentals of business process management* (2nd edition). Berlin: Springer.
- Enterprise Engineering Institute (n.d.). "Education." Retrieved 26 February 2020, 2020, from <https://demo.nl/>.
- Ettema, R. and J. L. Dietz (2009). Archimate and demo–mates to date? Advances in Enterprise Engineering III, Springer: 172-186.
- Figueira, C., & Aveiro, D. (2014). *A new action rule syntax for DEMO Models Based Automatic workflow procEss geneRation (DEMOBAKER)*. Cham, Springer International Publishing.
- Fink, A. (2005). *Conducting research literature reviews: From the Internet to paper* (2nd ed.). Thousand Oaks, Calif.: Sage Publications.
- Geskus, J. (2008). DEMO applied to Quality Management Systems, TU Delft, Electrical Engineering, Mathematics and Computer Science, Computer Science.
- Giachetti, R. E. (2010). *Design of enterprise systems: Theory, architecture, and methods*. Boca Raton, Fla.: CRC Press.
- Granho, J., Bakhshandeh, M., Pombinho, J., da Silva, M. M., & Caetano, A. (2014). *Validating value network business models by ontologies*. Paper presented at the Fourth International Symposium on Business Modeling and Software Design, BMSD.
- Gray, T., Bork, D., & De Vries, M. (2020). *A new DEMO modelling tool that facilitates model transformations*. Cham: Springer.
- Gray, T., & de Vries, M. (Accepted for Publication). *Empirical evaluation of a new DEMO modelling tool that facilitates model transformations* Paper presented at the 3rd International Workshop on Empirical Methods in Conceptual Modeling.
- Grigorova, K., Mironov, K. (2014). Comparison of business process modeling standards. *Int. J. of Eng. Sci. & Manag. Res.*, 1(3), Part 1-8.
- Guest, G., MacQueen, K. M., & Namey, E. E. (2011). *Applied thematic analysis*. Los Angeles: Sage Publications.
- Holt, J., & Perry, S. (2010). *Modelling Enterprise Architectures*. Institution of Engineering and Technology. Retrieved November 15, 2019, from <https://app.knovel.com/hotlink/toc/id:kpMEA00007/modelling-enterprise/modelling-enterprise>

- Kharmoum, N., El Bouchti, K., Ziti, S., & Omary, F. (2019). Descriptive analysis of business value models' transformation in MDA approach. *Proceedings of '3rd Scientific Days of Applied Sciences'*, 7(3), 71-78.
- Lara, P., Sánchez, M., & Villalobos, J. (2019). OT modeling: The enterprise beyond IT. *Business & Information Systems Engineering*, 61(4), 399-411.
- Leffingwell, D. (2011). *Agile software requirements: Lean requirements practices for teams, programs, and the enterprise*. Upper Saddle River, New Jersey: Addison-Wesley.
- Martinelli, R. J., & Milosevic, D. Z. (2016). 2.6 The alignment matrix. In *Project Management Toolbox* (2nd edition). John Wiley & Sons. Retrieved November 15, 2019, from <https://app.knovel.com/hotlink/pdf/id:kt011BDT52/project-management-toolbox/the-alignment-matrix>
- Mráz, O., Náplava, P., Pergl, R., & Skotnica, M. (2017). *Converting DEMO PSI transaction pattern into BPMN: A complete method*. Cham, Springer International Publishing.
- Mulder, M. A. T. (2019a). Towards a complete metamodel for DEMO CM. *Lecture notes in Computer Science (including subseries, Lecture notes in Artificial Intelligence and Lecture notes in Bioinformatics)*, 11231 LNCS, 97-106. doi:10.1007/978-3-030-11683-5_10
- Mulder, M. A. T. (2019b). *Validating the DEMO specification language*. Cham, Springer International Publishing.
- Mulder, M. A. T. (n.d.). *Enabling the automatic verification and exchange of DEMO models*. Netherlands (PhD).
- Nassar, V. (2012). Common criteria for usability review. *Work*, 41(Suppl 1), 1053-1057. doi:10.3233/WOR-2012-0282-1053
- Object Management Group. (2020). *Business process model & notation*. Retrieved 24 February, 2020, 2020, <https://www.omg.org/bpmn/>.
- Okoli, C., & Schabram, K. (2010). A guide to conducting a systematic literature review of information systems research, SSRN Electronic Journal.
- OMiLab. (n.d). *Why is ADOxx so efficient in developing modelling tools?* Retrieved 21 April, 2020, from <https://www.omilab.org/activities/adoxx.html>
- Peffer, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45-77.
- Perinforma, A. P. C. (2017). *The essence of organisation version 2.0: An introduction to enterprise engineering*, Sapio, www.sapio.nl
- Pombinho, J., Aveiro, D., & Tribolet, J. (2014). *A matching ontology for e3Value and DEMO: A sound bridging of business modelling and enterprise engineering*. Paper presented at the 2014 IEEE 16th Conference on Business Informatics (CBI) Geneva, Switzerland.
- Rodrigues, D. P. (2017). *Modelling business processes in BPMN inspired by the DEMO principles*. (Masters of Science Degree), Técnico Lisboa.
- Sauro, J., & Lewis, J. R. (2012). *Quantifying the user experience: Practical statistics for user research*, Retrieved 10 October, 2019, from <https://brad.idm.oclc.org/login?url=http://library.books24x7.com/library.asp?bookid=51045>

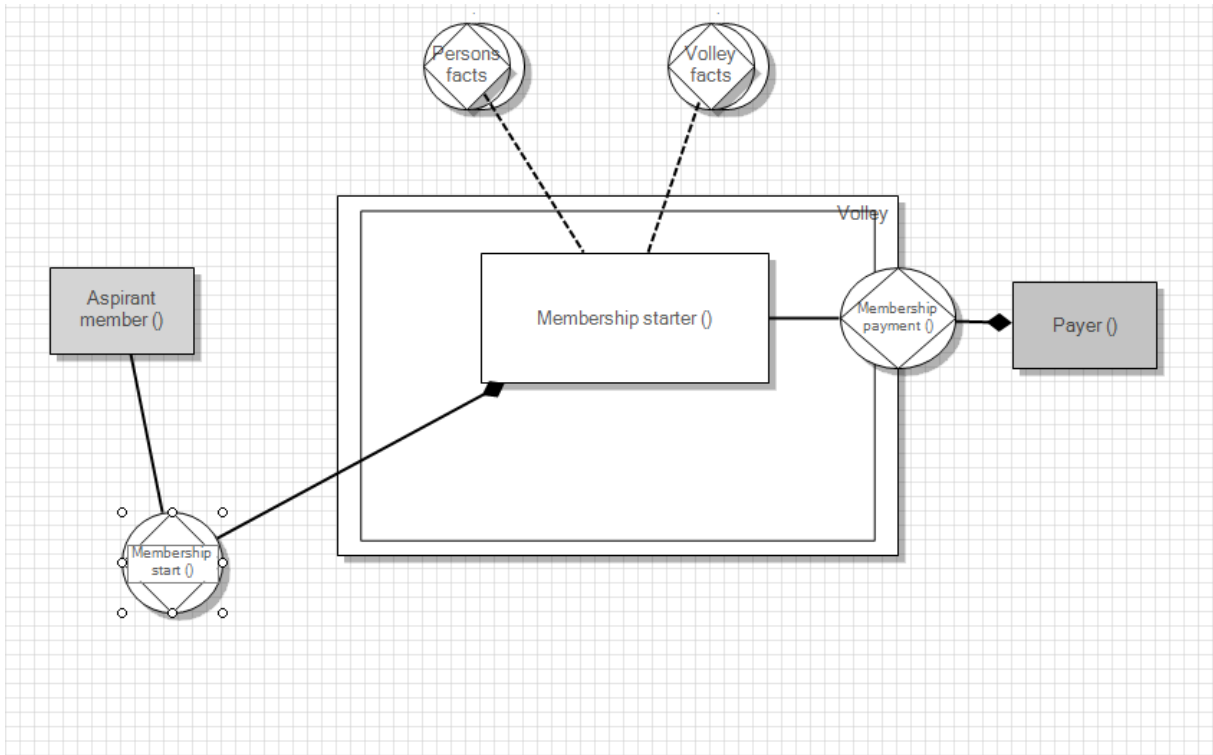
- Seck, M., & Barjis, J. (2015). An agent based approach for simulating DEMO enterprise models. *Procedia – Computer Science*, 61, 246-253. doi:10.1016/j.procs.2015.09.206
- Sein, M., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action design research. *Management Information Systems Quarterly*, 35(1), 37-56.
- Singh, P. M. (2013). *Integrating business value in enterprise architecture modeling and analysis*, (Masters thesis, University of Twente).
- Starr, M. (2013). Most useful keyboard shortcuts, Retrieved 10 October, 2019, 2019 from <https://www.cnet.com/news/most-useful-keyboard-shortcuts/>.
- Van Kervel, S. (2012). Ontology driven enterprise information systems engineering, Retrieved 10 October, 2019, 2019 from <https://doi.org/10.4233/uuid:8c42378a-8769-4a48-a7fb-f5457ede0759>
- Vejrazkova, Z., & Meshkat, A. (2013). *Translating DEMO models into Petri Net*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Von Rosing, M., Von Scheel, H., & Scheer, A. W. (2014). *The Complete Business Process Handbook: Body of Knowledge from Process Modeling to BPM, Volume 1 (Vol. 1)*. Morgan Kaufmann.
- Wang, Y., Albani, A., & Barjis, J. (2011). Transformation of DEMO metamodel into XML schema. In A. Albani, J. L. G. Dietz, & J. Verelst (Eds.), *EEWC 2011, LNBIP (Vol. 79, pp. 46-60)*. Heidelberg: Springer-Verlag.
- Wieringa, R. (2014). *Design science methodology for information systems and software engineering*. Heidelberg: Springer.
- Yamamoto, S., Olayan, N., & Morisaki, S. (2019). Analyzing e-Health business models using actor relationship matrix. *Acta Sci. Med. Sci*, 3(3), 105-111.
- Zachman, J. A. (1996). *The Framework for Enterprise Architecture: Background, Description and Utility*. Retrieved from http://www.acablogs.org/eakd/files/The_Framework_for_EA_Background_Description_and_Utility.pdf.

Chapter 10 Appendices

Appendix A

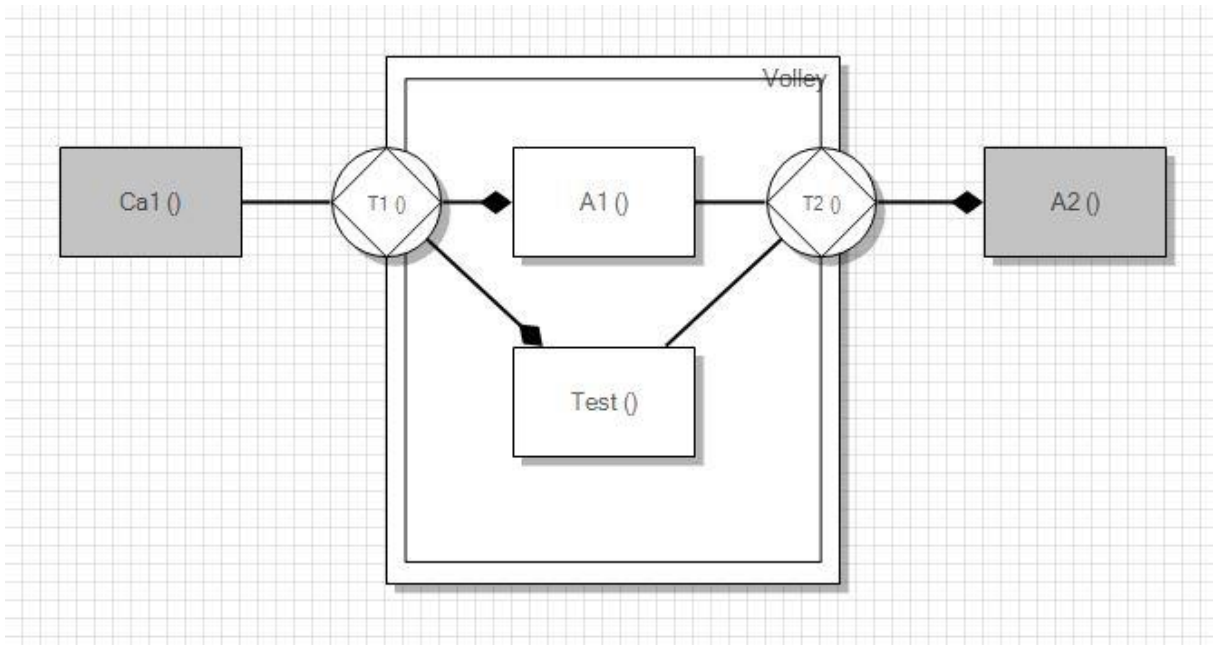


Abacus 1:

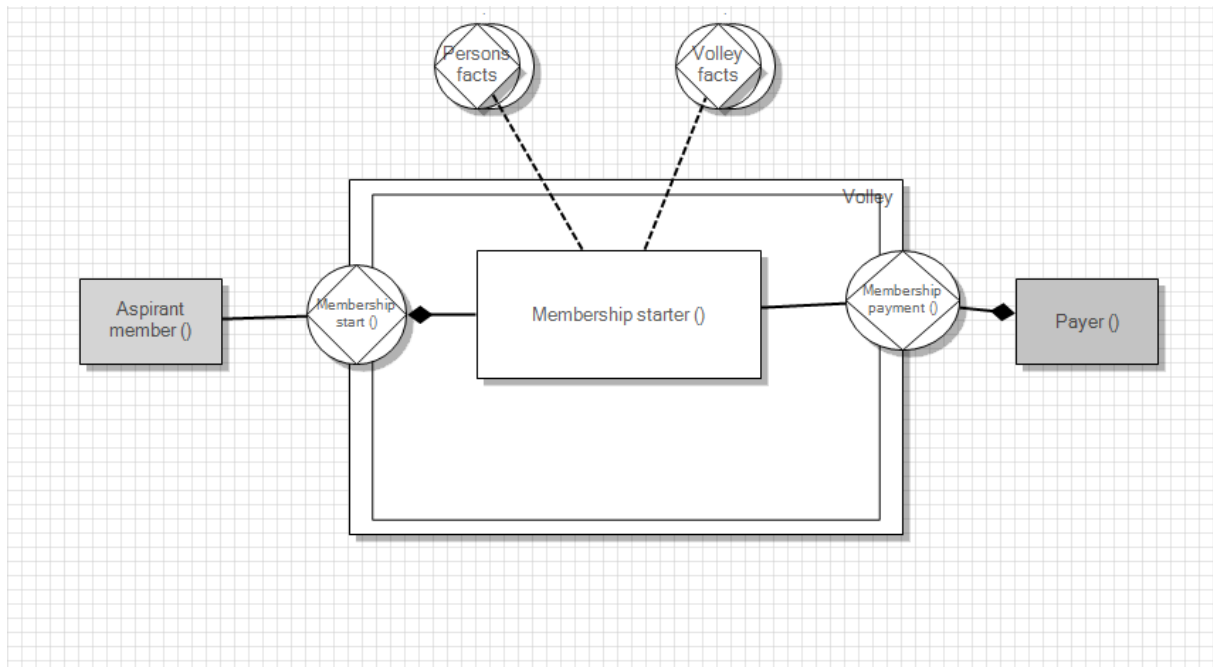


Abacus

2:



Abacus 3:



Abacus 4:

Model tree

- ▼ DEMO
 - ▼ aggregate production bank
 - A11 persons facts
 - A12 Volley facts
 - ▼ composite actor role
 - A4 aspirant member
 - A2 payer
 - ▼ elementary actor role
 - A1 membership starter
 - ▼ extension definition
 - external object class
 - object class
 - ▼ organizational boundary
 - Volley
 - ▼ transaction kind
 - I1 membership start
 - T2 membership payment

Active Model

My first model
properties | delete | new model

Diagrams in this model

DEMO construction diagram 1
diagram 1
diagram 2

properties | delete | new diagram

[Search the online repository](#)

Chat

share | print | download picture | report | export | import

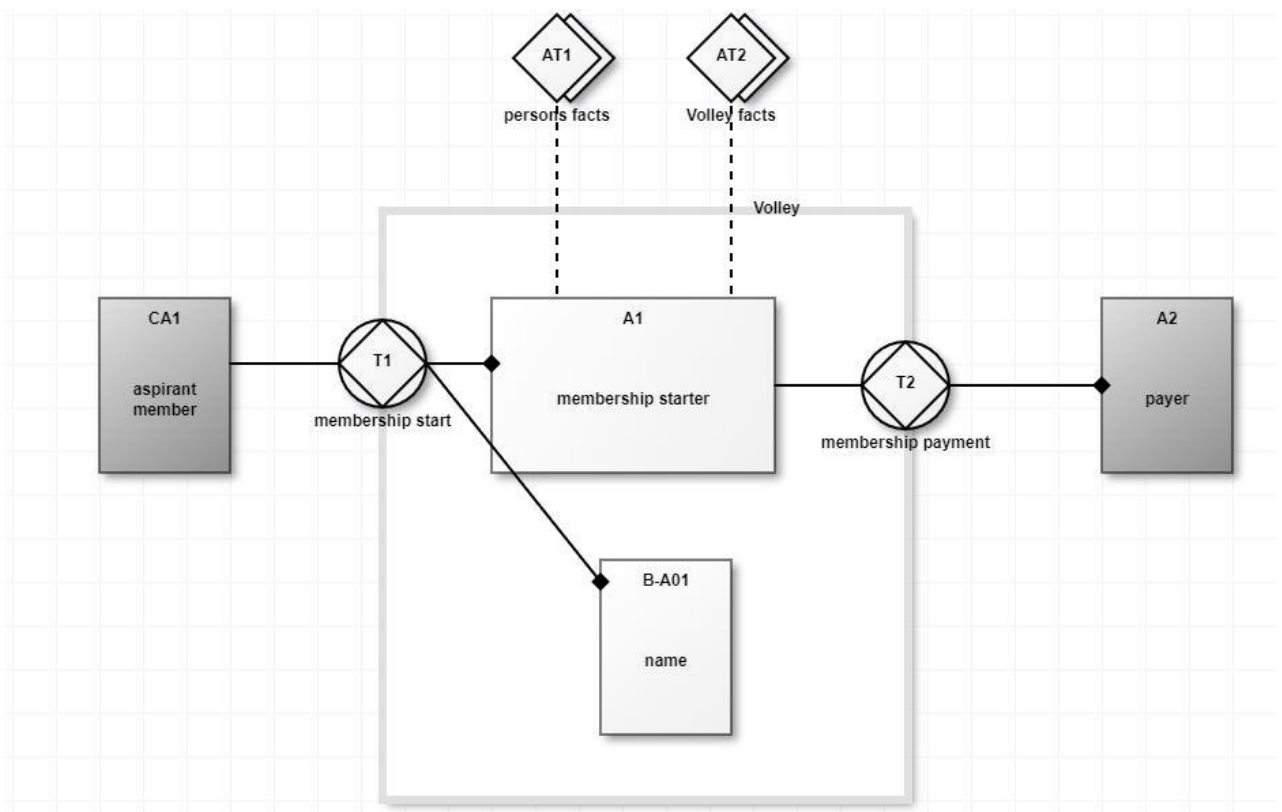
ModelWorld 1:

www.modelworld.nl says

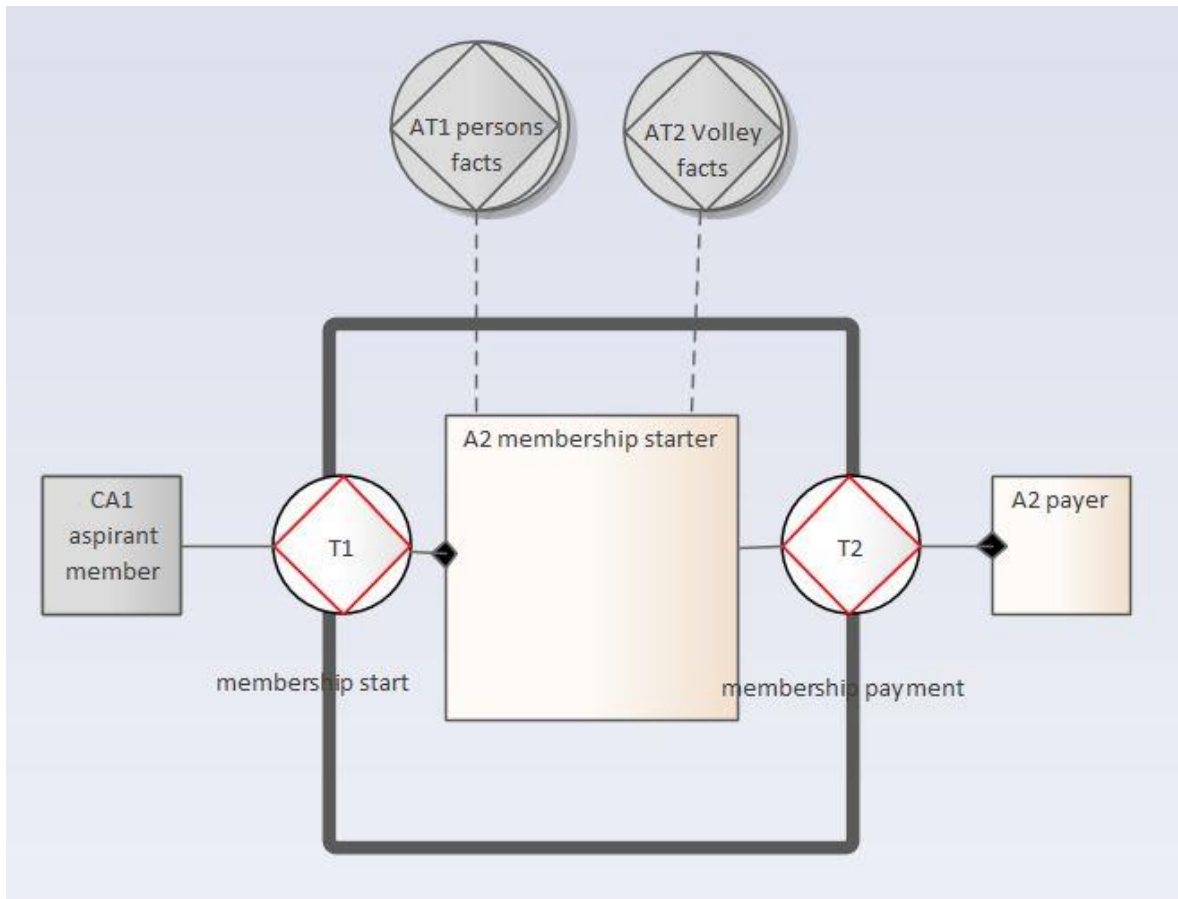
Cannot draw the elementary actor role here. It can only be drawn inside an organizational boundary.

OK

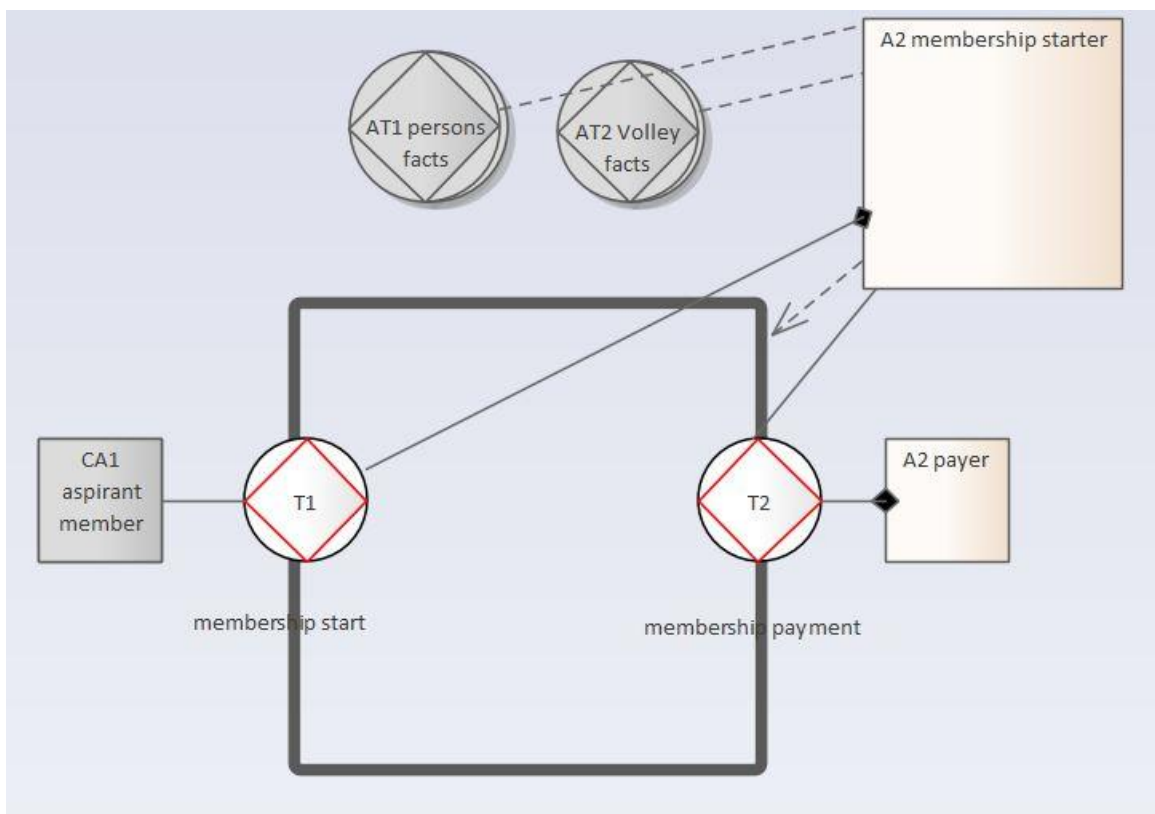
ModelWorld 2:



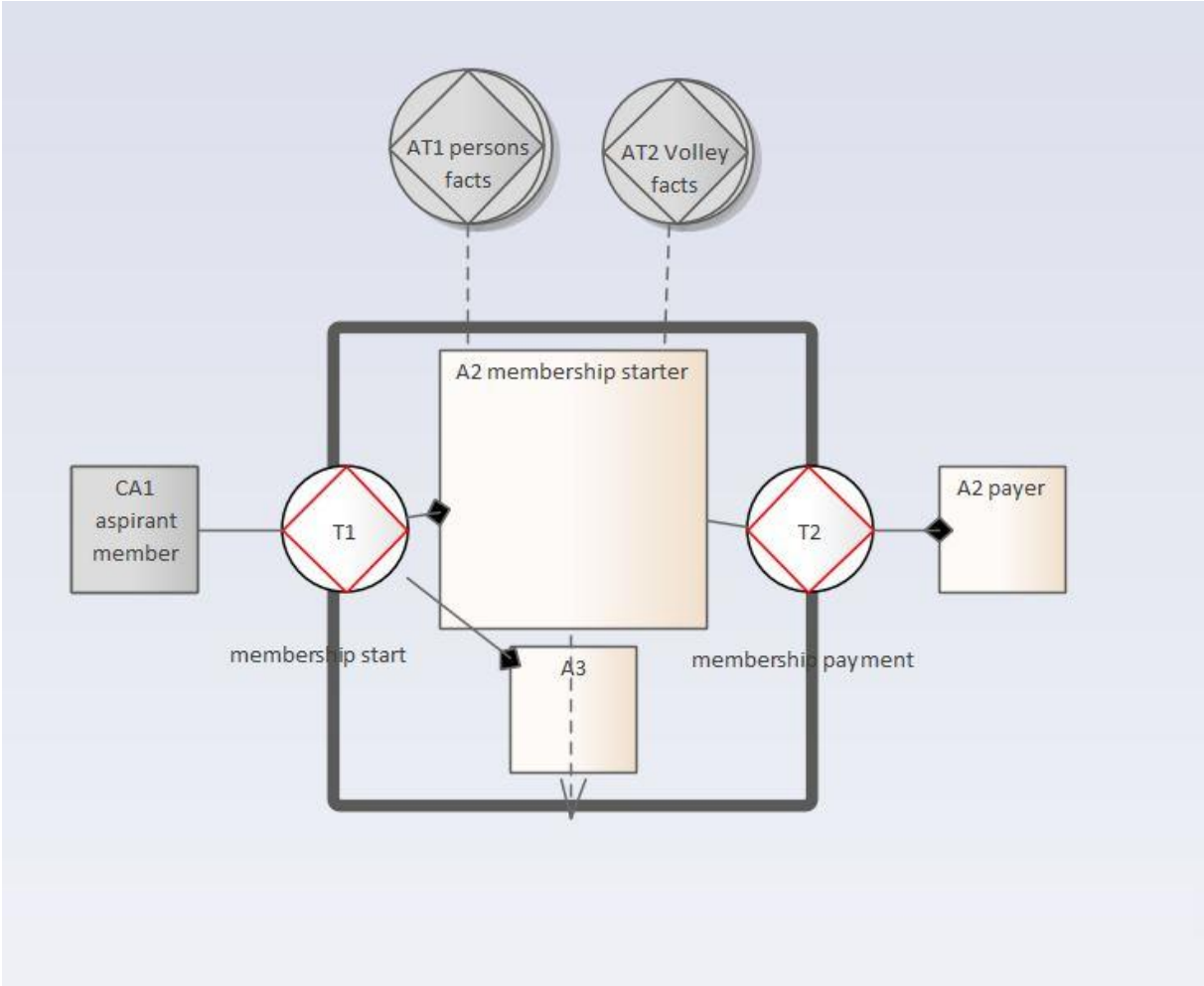
ModelWorld 4:



Plena 1:



Plena 2:



Plena 3:

The screenshot displays the Plena software interface, which is used for creating and managing UML diagrams. The main workspace shows a UML class diagram titled "Organisation Construction Diagram Package3".

Diagram Elements:

- Classes:**
 - CA1 aspirant member**: A class with a red border.
 - A2 membership starter**: A class with a yellow border.
 - A2 payer**: A class with a yellow border.
 - T1**: A class with a red border.
 - T2**: A class with a red border.
- Associations:**
 - membership start**: An association between CA1 aspirant member and A2 membership starter.
 - membership payment**: An association between A2 membership starter and A2 payer.
 - membership**: An association between A2 membership starter and T2.
 - membership**: An association between A2 membership starter and T1.
- Generalizations:**
 - A generalization relationship from CA1 aspirant member to A2 membership starter.
- Other Elements:**
 - AT1 persons' facts** and **AT2 Volley facts**: Two circular elements connected to the A2 membership starter class by dashed lines.

Interface Components:

- Top Bar:** Includes "Perspective" and "User" menus.
- Left Panel (Toolbox):** Contains various diagram elements such as "WorkInstruction", "WIS Links", "describes", "Common", "Note", "Constraint", "Text Element", "Diagram Legend", "Diagram Notes", "Hyperlink", "Artifact", "Requirement", "Issue", "Change", "Information Item", "Boundary", "Image", and "Common Relationships".
- Right Panel (Add-Ins):** Lists various tools and options like "Model Operators", "Element Creation", "Cache", and "Diagram Matrix".
- Bottom Panel (Find in Files):** A search bar for finding files, currently showing "File Search 1" and "Code Miner".
- Status Bar:** Shows "System Output", "Find in Files", and "Organisation Construction Diagram Package3: created: 2019/10/20 20:33:45 modified: 2019/11/14 11:08:52 100% 803 x 1146".

Plena 4:

Appendix B

👤 Coders

👤 Anthea ✖

👤 thoma ✖

➕ Add Coder

📁 All Documents

📁 New Semantic Domain ✖

➕ Add Code

📁 Semantic Domain: C1: Design Consistency, C2: Update Consistency, C3: DEMOSL External Consistency, C4: External Consistency... ✖

📁 C1: Design Consistency ✖

0x • 0 / 88 798 (0.0%)

📁 C2: Update Consistency ✖

3x • 366 / 88 798 (0.4%)

📁 C3: DEMOSL External Consistency ✖

1x • 189 / 88 798 (0.2%)

📁 C4: External Consistency ✖

1x • 105 / 88 798 (0.1%)

📁 C5: Metamodel inconsistency ✖

26x • 3 555 / 88 798 (4.0%)

📁 T1a: Conceptual Transformation ✖

6x • 854 / 88 798 (1.0%)

📁 T2: Transformation specifications ✖

0x • 0 / 88 798 (0.0%)

📁 T3: Direct Concept transformation ✖

0x • 0 / 88 798 (0.0%)

📁 T4: Transformation difficulty ✖

4x • 610 / 88 798 (0.7%)

➕ Add Code

Percent Agreement: 87.0%

➕ Add Semantic Domain

Percent Agreement: 87.0%

📁 D 1 [Mulder18] Validating the DEMO Specification Language ✖

📁 D 2 Bridging value modelling to ArchiMate via transaction modelling ✖

Appendix C

Research Question 5

Author	Title	Code	Relation to theme
<i>(Albani & Dietz, 2006)</i>	The benefit of enterprise ontology in identifying business components	CSA1	States that they have adopted consistency as a requirement for enterprise ontology, and that DEMO meets this requirement.
<i>(Badal, 2009)</i>	A roadmap for the transition from paper to digital records	CSA1	Mentions that the consistency between DEMO's models is naturally safeguarded.
<i>(Bakhshandeh, 2016)</i>	Ontology-driven analysis of enterprise architecture models	CSA1	Describes DEMO as being consistent with the constructivist paradigm.
<i>(Boedhram & Algoe, 2009)</i>	DEMO applied to financial services	CSA1	Applies DEMO to financial services, partially due to its consistency.
<i>(De Vries & Bork, In review)</i>	Transforming organisation design knowledge into executable business processes: An approach based on DEMO and BPMN (Unpublished)	CSA1	Mentions that DEMO's four aspect models present the organisation in a consistent way.
<i>(Ettema & Dietz, 2009)</i>	ArchiMate and DEMO – mates to date?	CSA1	States that the ontological model of DEMO is guaranteed to be consistent.
<i>(Geskus, 2008)</i>	Demo applied to quality management systems	CSA1	Describes DEMO models as claiming to be consistent, and then applies them to quality management systems.
<i>(Van Kervel, 2012)</i>	Ontology-driven enterprise information systems engineering	CSA1	The C4-ness claim is supported for DEMO in this study.
<i>(Mulder, 2019b)</i>	Validating the DEMO specification language	CSA2	According to this study, some inconsistencies in DEMOSL 3.6 were addressed in DEMOSL 3.7
<i>(Mulder, 2019b)</i>	Validating the DEMO specification language	CSA3	Proposes that DEMOSL 3.7 be extended to describe all the necessary model aspects.
<i>(Mulder, 2019b)</i>	Validating the DEMO specification language	CSA4	Refers to the meta-model that (Van Kervel, 2012) proposed.

<i>(Van Kervel, 2012)</i>	Ontology-driven enterprise information systems engineering	CSA4	Claims that DMOL, the meta-model proposed by the author, satisfies the C4 criteria.
<i>(Wang, 2009)</i>	Transformation of demo models into exchangeable format	CSA4	Claims that the meta-model described is a complete representation of all models.

Research Question 7

<i>(Bakhshandeh, 2016)</i>	Ontology-driven analysis of enterprise architecture models	TSA1	Refers to the proposed transformation (de Kinderen et al., 2014); does not use it further.
<i>(De Vries & Bork, In review)</i>	Transforming organisation design knowledge into executable business processes: An approach based on DEMO and BPMN (in review)	TSA1	Proposes a DEMO to BPMN transformation, with very explicit specification about how the concepts within the meta-models are related.
<i>(Ding, 2016)</i>	Integrating value modeling into ArchiMate	TSA1	Refers to the proposed transformation (de Kinderen et al., 2014); does not use it further.
<i>(de Kinderen et al., 2014)</i>	Bridging value modelling to ArchiMate via transaction modelling	TSA1	Proposes transforming DEMO to ArchiMate by performing class-to-class and relation-to-relation concept mapping between the models.
<i>(Granjo et al., 2014)</i>	Validating value network business models by ontologies	TSA1	Refers to the proposed transformations (de Kinderen et al., 2014) and (Pombinho et al., 2014); does not use them further.
<i>(Figueira & Aveiro, 2014)</i>	A new action rule syntax for DEMO MODELS Based Automatic workflow procEss geneRation	TSA1	States that some concepts can be mapped from the DEMO AM to a BPMN model.
<i>(Kharmoum et al., 2019)</i>	Descriptive analysis of business value models' transformation in MDA approach	TSA1	Refers to the proposed transformation (de Kinderen et al., 2014); does not use it further.
<i>(Lara et al., 2019)</i>	OT modeling: The enterprise beyond IT	TSA1	Refers to the proposed transformation (de Kinderen et al., 2014); does not use it further.
<i>(Mráz et al., 2017)</i>	Converting DEMO PSI transaction pattern into BPMN: A complete method	TSA1	Proposes a top-down approach to use DEMO models to generate BPMN models. They provide some specifications for this transformation.

<i>(Pombinho et al., 2014)</i>	A matching ontology for e3Value and DEMO – A sound bridging of business modelling and enterprise engineering	TSA1	Proposes using essential matching of concepts between the e3Value and DEMO meta-models to ensure consistency between models. Does not perform transformation.
<i>(Singh, 2013)</i>	Integrating business value in enterprise architecture modeling and analysis	TSA1	Refers to the proposed transformation (de Kinderen et al., 2014); does not use it further.
<i>(Wang, 2009)</i>	Transformation of demo models into exchangeable format	TSA1	DEMO is transformed into XML by identifying which DEMO elements correspond to which XML elements.
<i>(Wang et al., 2011)</i>	Transformation of DEMO metamodel into XML schema	TSA1	DEMO is transformed into XML by mapping similar concepts from DEMO's meta-model to XML documents.
<i>(Yamamoto et al., 2019)</i>	Analyzing e-Health business models using actor relationship matrix	TSA1	Refers to the proposed transformation (de Kinderen et al., 2014); does not use it further.
<i>(Vejrazkova & Meshkat, 2013)</i>	Translating DEMO models into Petri Net	TSA1	Proposes the creation of a set of modelling constructs to facilitate building Petri models from DEMO models.
<i>(Seck & Barjis, 2015)</i>	An agent based approach for simulating DEMO enterprise models	TSA2	DEMO is translated into an agent-based simulation by considering the actors and transactions in a specific instance of the DEMO model.