# MASTER'S THESIS

## A method of consistent Enterprise Modeling in ArchiMate

Wols, M.

**Award date:**
2022

# A method of consistent Enterprise Modeling in ArchiMate

| | |
|---|---|
| Degree programme: | Open University of the Netherlands, Faculty Science |
| | Master of Science Business Process Management & IT |
| Course: | IM0602 BPMIT Graduation Assignment Preparation<br>IM9806 Business Process Management and IT Graduation Assignment |
| Student: | Marco Wols |
| Identification number: | Student Number |
| Date: | June 7, 2022 |
| Thesis supervisor | Dr. Ir. Ella Roubtsova |
| Second reader | Dr. Ben Roelens |
| Version number: | 1 |
| Status: | final |

# Abstract

In Enterprise Modeling, the enterprise is modeled from different perspectives. Those perspectives, often called views, should be consistent, i.e. the same element and relation of the enterprise should be represented in the different views with the same names. The modeling language ArchiMate was designed to increase consistency between enterprise models created in other languages. It uses a single internal model to relate the elements of the different views. When using ArchiMate for enterprise modeling however, still the potential of inconsistencies in the enterprise model exists, as consistency requires a structural modeling approach. With the use of the modeling approach of two other modeling languages, a modeling method is proposed to create a consistent enterprise model in ArchiMate. This method refines goals to requirements and derives the elements and relations from these requirements by lexical analysis. Those elements and relations can be used to create the other views via specific rules. Using laboratory testing on a theoretical case and field testing on a real-life case, the proposed method is tested.



# Key terms

Enterprise Modeling, ArchiMate, Elements and relations of the internal model, Motivation aspect, Consistency.

# Contents

# 1. Introduction

## 1.1. Background

Enterprise Modeling (EM) is a modeling approach which allows to look at a chosen business process or aspect of an enterprise from different perspectives. In such multi-view modeling, there is a potential of inconsistencies between views. In the modeling language UML for example, this relates among others to elements that are defined in a view, but used differently in other views (Roubtsova, van Gool, Kuiper, & Jonkers, 2002). Also missing definitions of elements and unused defined elements can be seen as inconsistencies of the complete model. For EM, comparable problems occur when elements are used with different names or different meaning in different views. The consistency is even more challenging when views have different semantics where e.g. a relation in a concepts view, represents a business process in a behavior view.

ArchiMate ("ArchiMate® 3.1 Specification," 2019) is one of the frequently used modeling languages in EM. It is a standard from The Open Group. ArchiMate uses an internal model containing all elements and relations. When this internal model has been filled in with unique (non-duplicating) elements, this can be used as a source for creating views that are consistent with each other. There is however a problem to fill the internal model with unique (non-duplicating) elements, because the choosing and naming of elements is done manually, which can lead to the use of different names for the same elements and relations, and may use not related elements etc. This research is about a modeling method in ArchiMate to fill the internal model with unique elements and create consistent views, to assure the consistency of an Enterprise Model in ArchiMate.

Various EM languages use slightly different terms for the same concepts of their language. Especially the word ‘concept’ has different definitions and usage. To avoid confusion on the exact meaning of specific terms, this research uses the terms defined within the ArchiMate specification ("ArchiMate® 3.1 Specification," 2019), unless described otherwise.

## 1.2. Exploration of the topic

Inconsistency in Enterprise Models within ArchiMate has various appearances and causes. Let us illustrate an inconsistency using a case study Library services. This case study is textually taken from Roubtsova (2016) and reprinted with permission in Appendix 1. The case has originally been modeled using protocol models in an EM approach called ExtREME (Executable Requirements Engineering, Modeling and Evolution). In this work this case has been modeled in ArchiMate. The naming and the choice of modeling constructions are new.

Figure 1 shows the services that a library delivers to the customer. The element Customer is a business actor. The element Customer Services is a business service which is composed of five business services on a more concrete level.

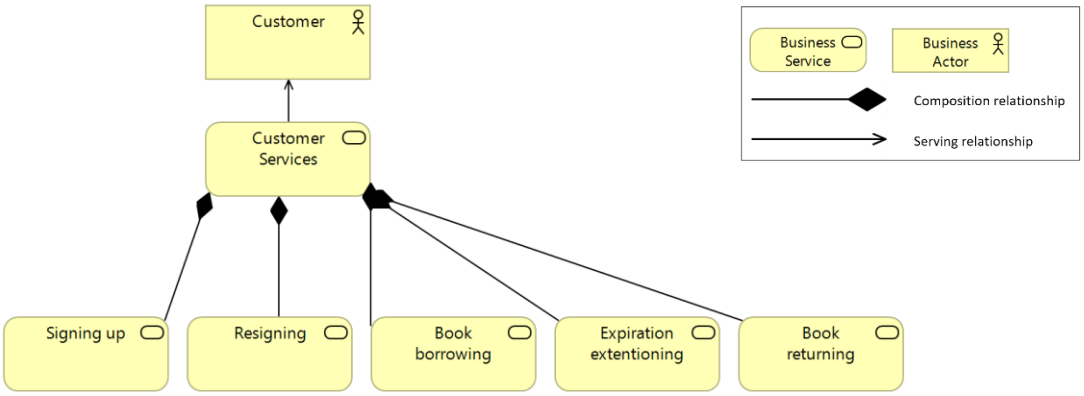


*Figure 1: View of the external services of a library*

Figure 2 shows the business processes (e.g. Signing up process) depicted as a group containing an event and process steps. The event triggers the process steps and each process step triggers the next. The OR junction (relationship connector) specifies in these cases exclusive-OR conditions as choices in the process flow. Those business processes realize the service the library delivers to the customer.

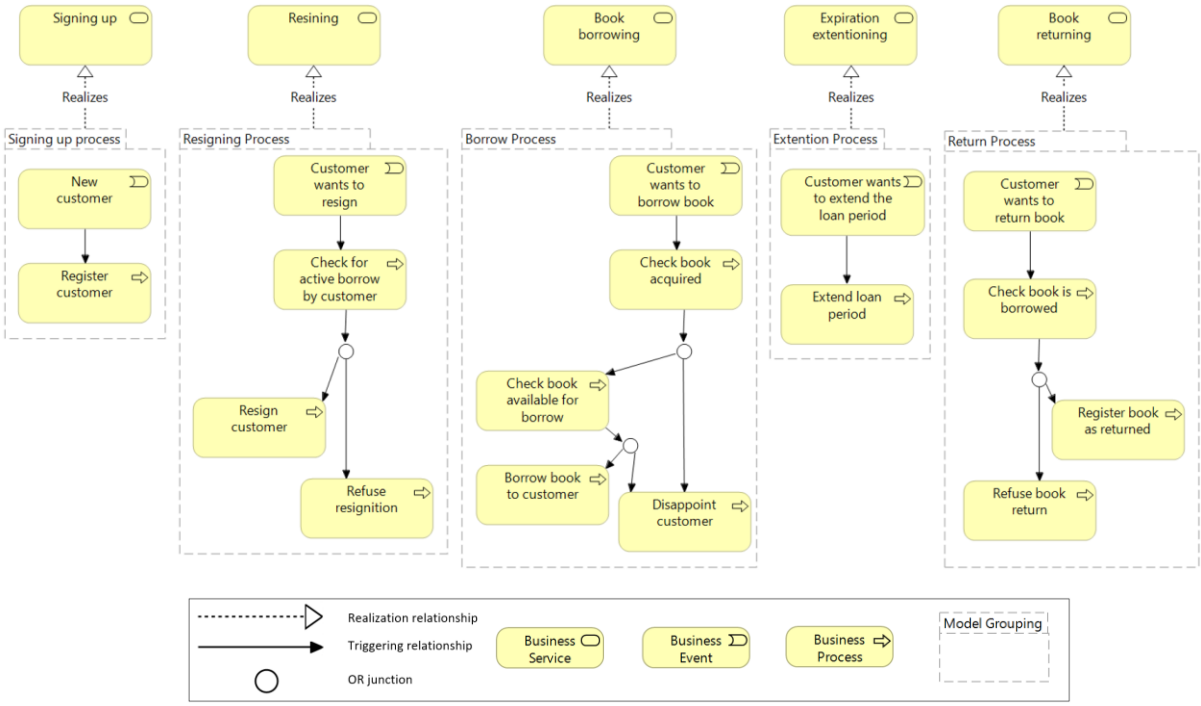


*Figure 2: View of the behavior elements related to the customer services*

In Figure 3, snippets of the internal model of ArchiMate are shown. The ‘Signing up’ service only exists once in the model; the visualizations in both views are derived from the same element. On the contrary, the service ‘Book borrowing’ has two instances in the model, each created in one of the views. Because these two elements describe the same aspect of the enterprise, this is an inconsistency in the model. A comparable problem is there with the ‘Resigning’ service. It also has two instances, but they have different names due to a typo.

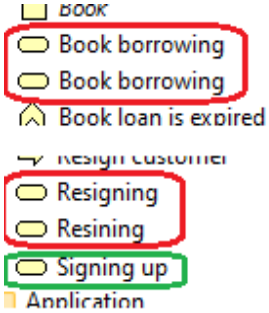


*Figure 3: Snippets from the library model*

This example illustrates the appearance of inconsistencies in ArchiMate models because of typos, because of thinking and rethinking and making decisions about modeling constructions during modeling.

## 1.3. Problem statement

An enterprise model in ArchiMate should be a precise representation of a chosen business case, a chosen business aspect or a business process in the enterprise. A case description is modeled as views and it results in an internal set of modeling elements and their relations in ArchiMate. It is difficult to ensure that the internal set of elements reflects the case description as many views contribute to this internal set without any fixed method of modeling of views and controls of consistency.
EM is often a process of constant expansion, adaptation and rearrangement of the internal set via views. However, there is no finite iterative method for this: Directing the enterprise model to the state of testable correspondence with the case description.

## 1.4. Research questions and main line of approach

From the research problem the following main research question was formulated:

> *What can be a method of multi-view enterprise modeling in ArchiMate, that can finitely direct the model to the state of testable correspondence with the case description?*

To answer this main research question, the following detailed research questions are defined. For each research question the method is described to answer this question.

1. *What are the basics of Enterprise Modeling in ArchiMate? How is consistency managed between views and what consistency issues can occur within ArchiMate?*

   For this research question, the document analysis method is used and the ArchiMate specification ("ArchiMate® 3.1 Specification," 2019) is analyzed. With the use of the Archi modeling tool (Archi, 2021), examples of ArchiMate models are created to substantiate the findings. Also additional literature will be searched on the concepts behind the development of ArchiMate.

2. *What modeling methods are used in other EM approaches/ languages? What requirements are there for a good EM modeling method? Is there a method for enterprise modeling in ArchiMate, that can finitely direct the model to the state of testable correspondence with the case description?*

For this a literature analysis is conducted in which 2 other EM approached are examined. Also the article of Ralyté, Bork, Jeusfeld, Kirikova, and Stirna (2021) is used to understand the requirements for a 'perfect' Enterprise Modeling Method. With this information and the understanding of ArchiMate from research question 1, a method is proposed for modeling in ArchiMate to create consistent models. This model is developed using concepts of Design Science Research (DSR) (Hevner, 2007).

3. *How does the proposed method for modeling in ArchiMate, to create EA models, works in practice? Are there restrictions of its application?*

   For this research question, a case study is conducted as part of the DSR-approach.

## 1.5. Motivation/relevance

ArchiMate is intended as a EM modeling language for "bringing together already existing techniques and integrating these at the appropriate level of abstraction" (Lankhorst, 2004). To do so, such a language should globally model the structure within each domain and model the relations between the domains. By allowing to model dependencies between the different layers, domains and views, Lankhorst, Proper, and Jonkers (2010) state that an ArchiMate model "becomes a coherent whole instead of a collection of isolated diagrams of different kinds". As such, it is important that the models within ArchiMate themselves are consistent.

Although ArchiMate is a frequently used modeling language, there is, to the best of the authors' knowledge, no method available to achieve a consistent model within ArchiMate, using a defined set of steps.

# 2. Theoretical background

## 2.1. Research method for literature analysis

As a starting point of the literature analysis, several literature directions were advised by the author's supervisor, including the ArchiMate Specification retrieved from the Open Group website ("ArchiMate® 3.1 Specification," 2019), background on the 4EM method (Sandkuhl, Stirna, Persson, & Wißotzki, 2014) and the ExtREME method (Roubtsova, 2016).

To get more information on the concepts and background of ArchiMate and its use, a literature search was conducted using the advanced search function in the OU library ("OU Library - Advanced search,"). Initially a search was done for all available libraries, publication dates 01-01-2014 till 01-02-2022, on titles containing 'ArchiMate'. This search gave, counting only peer reviewed publications, 21 results. Those publications concerned mainly the usage of ArchiMate, discussed additions etc. Back searching the references of these publications, ArchiMate turned out originating from an earlier period with a 1.0 specification as from 2009. Resubmitting the search for the period 2000 – 2014 gave 2 results, including a paper where Lankhorst, Proper and Jonkers present the ArchiMate Language (Lankhorst et al., 2010). This paper and some of its references were used to study the background and concepts of ArchiMate.

Another search was used to find publications on consistent modeling methods with ArchiMate. This search filters on peer-reviews publications with 'ArchiMate' in the title and containing in any field either 'modeling' or 'model' and 'consistency' or 'consistent'. Searching from 01-01-2000 till 01-02-2022. This resulted in 8 publications, including the already found (Lankhorst et al., 2010), from which no publication describes a modeling approach to create consistent enterprise models in ArchiMate from a case description.

Additional literature was found by snowball searches and occasional searches made on specific topics, e.g. on the KAOS method by Dardenne, Van Lamsweerde, and Fickas (1993).

## 2.2. Analysis of the ArchiMate specification to understand reasons of possible inconsistencies

The ArchiMate Specification (ArchiMate 3.1., 2019) is analyzed to answer the first research question. To improve readability, shortened references are used to the ArchiMate Specification by using the notation (AMS, §<paragraph number>). Using the Archi tool (Archi, 2021), examples are created to test and substantiate the findings from the analysis. As basis and inspiration for the modeling experiments, the Library example is used.

### 2.2.1. Model, Views and Viewpoints in ArchiMate

ArchiMate uses a single internal model for EM. This internal model in ArchiMate consists of a set of elements, relationships, and relationship connectors (AMS, §2.14 and §2.8). However this set is in practice created by drawing views. Each view is only a partial presentation of the case description. ArchiMate uses the term architectural view or simply view, as the actual views in the model. Those are described as "A representation of a system from the perspective of a related set of concerns". (AMS, §2.3). The views in ArchiMate are not specifically prescribed and can be any subset of the

elements from the internal model and the relations between those elements. Still the views are meant to give information about certain concerns of one or more stakeholders.

To frame one or more concerns of a stakeholder, viewpoints are used in ArchiMate (AMS, §14.3). An architectural viewpoint or simply viewpoint is defined as “A specification of the conventions for a particular kind of architecture view” (AMS, §2.4). Those viewpoints prescribe a set of element types which are allowed in a view that is in line with that viewpoint. In Archi, the viewpoints act as a filter on the elements that can be added to a view. As an example the Organization Viewpoint is taken defined in (AMS, §C.1.1.). The element types of this viewpoint are listed in Table 1.

*Table 1: Element types of the Organization Viewpoint. Source: (AMS, §C.1.1)*

| **Element Type** |
|---|
| Business Actor |
| Business Role |
| Business collaboration |
| Location |
| Business Interface |

When using the Organization Viewpoint on a specific view in the Archi tool, the subset of elements in the Pallet pane is reduced to the elements belonging to that viewpoint. This is shown in Figure 4.

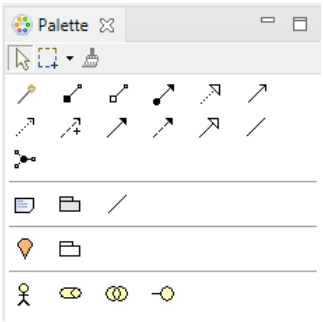


*Figure 4: Pallet in Archi tool when using Organization Viewpoint*

The view itself shows the elements not belonging to the viewpoint but already in the view, as greyed-out (Figure 5). Note that this specific functionality where views can be graphically filtered is called ‘Dynamic Viewpoints’, and is not in the ArchiMate specification but a specific feature of the Archi tool ("Archi User Guide," 2021).

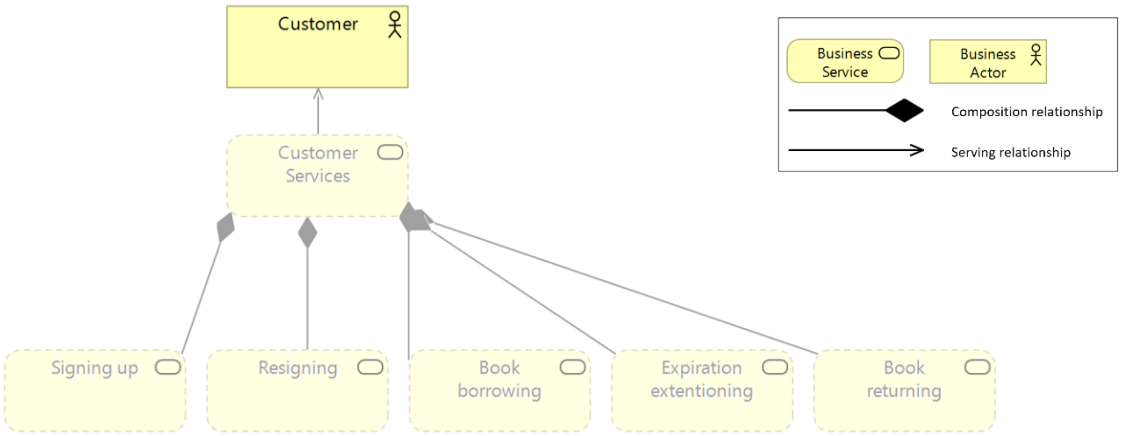


*Figure 5: Business Services view when using Organization Viewpoint*

Figure 4 shows that the choice of a viewpoint does not limit the relations between elements as all relations can still be chosen in het Pallet pane. If the element types on both sides of the relation are defined by the viewpoint, the relation can also be shown in the view.

ArchiMate allows viewpoints to be defined as every subset of elements. Not all viewpoints would however give meaningful results. Based on common architectural practice and on experiences with the use of ArchiMate models in practical cases, useful combinations in the form of a set of basic viewpoints have been defined (AMS, §C.1). In (AMS; Chapter C), 25 example viewpoint are defined. In the definitions, however, no specific semantics are described. The viewpoint only defines the selection of the elements which are allowed in a view, as subset of all elements available in ArchiMate. As the view is just a display of a part of the total model, the meaning and rules for the usage of the ArchiMate elements and relations do not differ between the views. In that way the views in ArchiMate do not have different semantics and present a mixture of elements and relations on potentially different aspects and layers.

### 2.2.2. Different graphical notations

In ArchiMate, different graphical notations are allowed to express structural relations (AMS, §5.1). Figure 6 shows two examples (AMS, §5.1.1), representing the composition relationship. The right diagram is however ambiguous as it could also visualize e.g. an aggregation relationship.

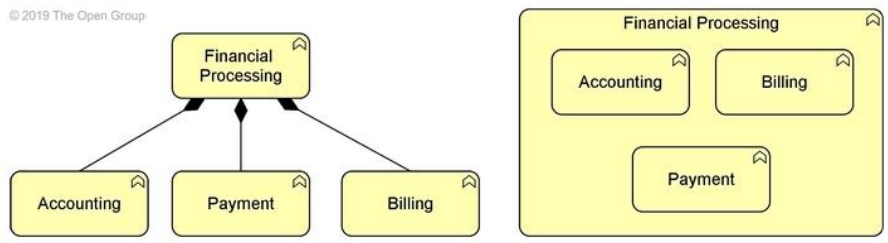


*Figure 6: Different graphical notations for structural relationships. Source: (AMS, §5.1.1)*

The Archi tool will however ask for the concerning relation when inserting an icon in another icon (Figure 7). This results in an internal model which still represents the modelers' intentions (Figure 8).

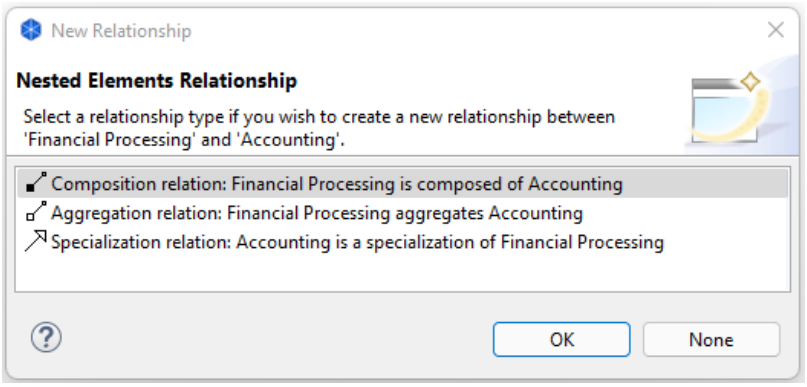


*Figure 7: Archi tool inquires element relations*

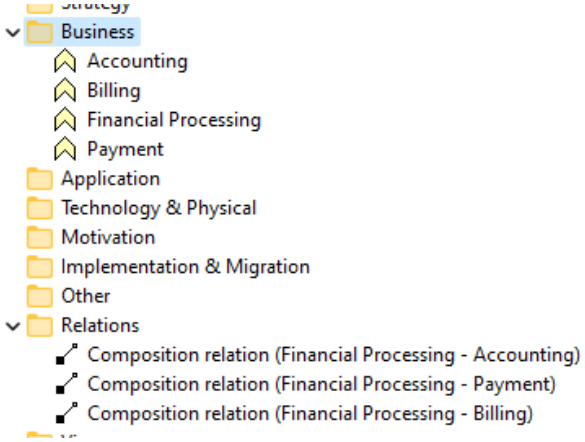


*Figure 8: Internal model when using icons in icon diagram*

### 2.2.3. Modeling abstractions

Figure 9 shows the core framework of ArchiMate (AMS, §3.5), in which three main layers have been distinguished. Those layers represent different abstractions of the same enterprise. The available elements in ArchiMate vary per layer. In some cases the same type of element exists for the different layers, having a refinement for each layer. For instance, Business Service vs. Application Service vs. Technology Service. In other cases an element only exists in a specific layer, e.g. an Actor only exists in the Business Layer.

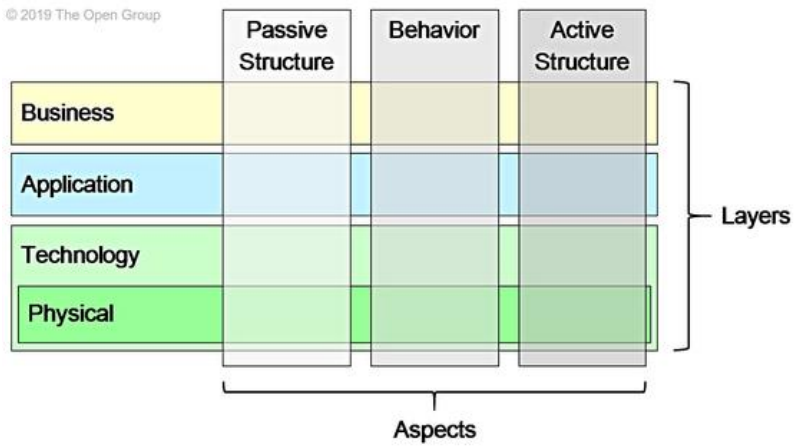


*Figure 9: The ArchiMate core framework. Source: (AMS, §3.1)*

In the current version of ArchiMate, extensions are added to the core framework. These extensions consist of two additional layers that model the Strategy and the Implementation & Migration (Figure 10). (AMS, Chapter 7) states that "The strategy elements are typically used to model the strategic direction and choices of an enterprise, as far as the impact on its architecture is concerned". The Implementation & Migration elements "support the implementation and migration of architectures" (AMS, Chapter 13) and therefore could be used for the transition roadmap (Kotusev, 2019) as part of a complete enterprise architecture, additional to the AS-IS and TO-BE models.

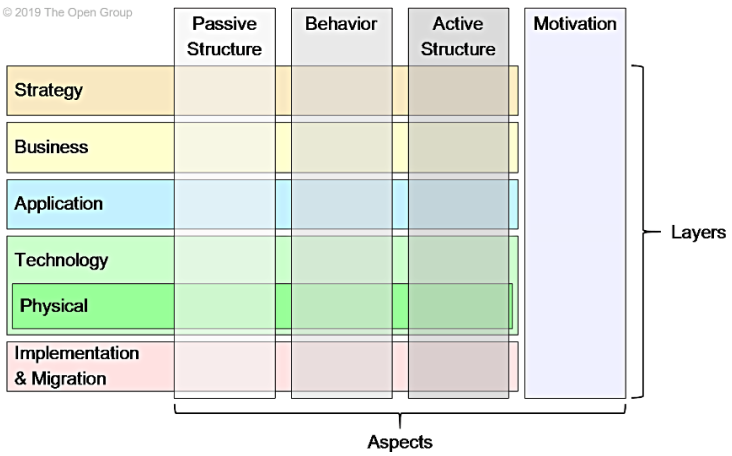


*Figure 10: The ArchiMate full framework. Source: (AMS, §3.5)*

When modeling in ArchiMate, consideration should be given to the layers that will be used in the model. Sandkuhl et al. (2014) state "the task, and therefore the problem to be solved, must be clearly defined and should form the basis for defining the purpose of the model". This research uses examples on the business layer, but when modeling applications or modeling for simulation, the application layer might be more appropriate, as the application layer "depicts application services that support the business, and the applications that realize them" (AMS, §3.3).

### 2.2.4. Modeling of core elements in ArchiMate

The core elements of ArchiMate are the elements on the core layers of the ArchiMate framework and are divided into three aspects: Passive Structure, Behavior and Active Structure. The complete core model relates elements of those three aspects. The general approach is that passive elements are altered/ used by behavior. The behavior is executed by active elements. However, there are also specific composite elements that do not fit in one aspect (AMS, §3.4).

ArchiMate distinguishes three groups of relationships to model the relations between het elements:

1. Structural Relationships which represent the "static" coherence within an Architecture (AMS, §5.1), e.g. composition and realization.
2. Dependency Relationships describe how elements support or are used by other elements (AMS §5.2), e.g. serving, access or influence.
3. Dynamic Relationships describe temporal dependencies between elements, e.g. triggering and flow (AMS, §5.3).

Note that none of the defined relationships in ArchiMate represent a relation of behavior. The Association relationship, as special case of the Dependency Relationships, might be used for "an unspecified relationship, or one that is not represented by another ArchiMate relationship" (AMS,

§5.2.4). However, in ArchiMate, behavior is modeled with behavior elements (AMS, §4.1.2), not as relations between other elements. This results in a single definition for behavior in the model.

Outside ArchiMate, there are several families of approaches on behavior modeling. Approaches originating from computer science (Roubtsova, 2015), are often variants of the Finite State Machine (Belzer, Holzman, & Kent, 1975) modeling states, events and transitions. This is a common approach for models of computer programs and simulations, requiring stringent state definitions and transitions. Approaches based on Business Management on the other hand, model business processes as a sequence of activities, not necessarily including all possible exceptions and details. Those processes are often modeled on multiple levels of abstraction, each with its own purpose (Dumas, La Rosa, Mendling, & Reijers, 2018). Those models are often used in Business Management when focusing on certain process measures and performance (e.g. lead time). The ArchiMate behavior modeling has however no capabilities to do detailed process modeling in both families of approaches. For that kind of detailed modeling, other business process design languages such as BPMN should be used (AMS, §8.3.1). ArchiMate has behavior elements for processes, events and functions on the various layers. ArchiMate has no elements to model states, therefore behavior modeling based on Business Management approaches is the most logical option in ArchiMate. A basic version of behavior modeling using state machines could however be depicted by using the event element connected with triggers because “a business event represents an organizational state change” (AMS, §8.3.4).

### 2.2.5. Motivation modeling in ArchiMate

The motivation aspect is one of the additions to the core framework of ArchiMate (AMS, §3.5). Figure 10 illustrates that this aspect does not specializes for the layers but exists fully besides those layers. Where the other aspects and layers describe the architecture of the enterprise, the motivation aspect describes the “why” of this architecture being the context or reason behind the architecture (AMS, §4.4). As such, the motivation aspect represents the case description of the enterprise.

The motivation aspect in ArchiMate uses various elements. The core of these elements describe the goals of the enterprise, refined at various levels and with specific meaning: Goal, Outcome, Principle, Requirement and Constraint (AMS, §6.3). Besides those elements, other elements exist to create the context and the why behind the goals: Stakeholder, Driver and Assessment (AMS, §6.2). ArchiMate even has elements to model for the different interests of different stakeholders using the Meaning an Value elements (AMS, §6.4).

As “a requirement represents a statement of need defining a property that applies to a specific system as described by the architecture” (AMS, §6.3.4), the requirements can be used to derive the required elements in the core model.

## 2.3. Discussion of potential inconsistency issues in ArchiMate models

ArchiMate uses an internal model on which the various views are based upon. As a stakeholder will use the views to get informed on his concerns of the enterprise, those views should be consistent with each other and the internal model. This paragraph describes some examples of inconsistencies that might be created within an ArchiMate model and its views during modeling.

1. In §1.2 of this document, potential inconsistencies are already shown regarding double elements, whether or not caused by typos. This situation is caused by creating elements within each view separately. It will not occur when creating the elements in the internal model or in only one view, and referring to them in other views. In Archi, such a reference can be made by dragging the element from the internal model onto the view. However no elements should be created in the internal model, which are not used in views. Note that the tool Archi has a validator tool that checks the internal model for amongst others unused elements and duplicate names for the same element type ("Archi User Guide," 2021).

2. Another potential inconsistency problem arises from the fact that ArchiMate does not enforce all elements of a specific type to be in a view, even if this view is based upon a specific viewpoint. Although there is a good reason in case of large models that need splitting around certain subjects, a stakeholder might not know about the elements that are missing in a view. A special case however is the situation where a composition relation is used as shown in Figure 1. When deleting the element Book Returning from this view as depicted in Figure 11, the internal model still holds this element and the composition relation that is between this element and Customer Services (Figure 12). With that, the view suggests a composition of Customer Services of only 4 elements, where the internal model has a composition of 5 elements.

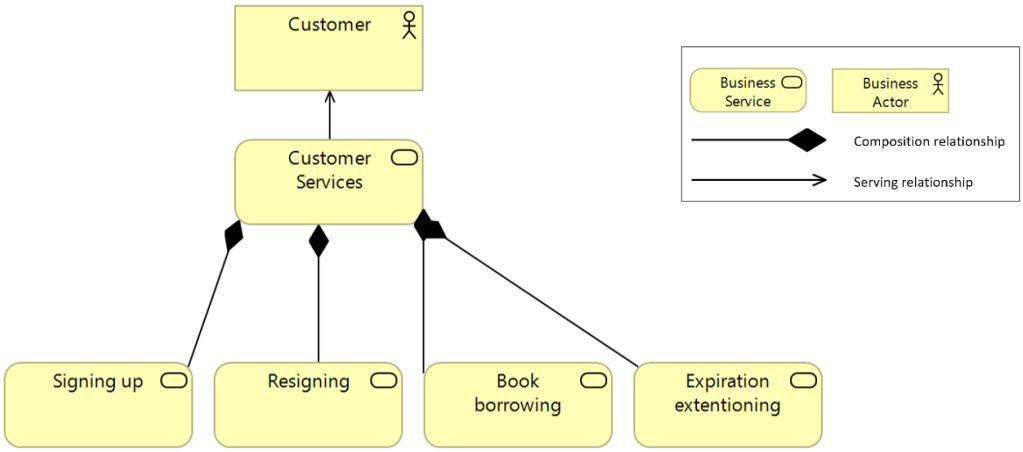


*Figure 11: Incomplete view on composition of customer services*

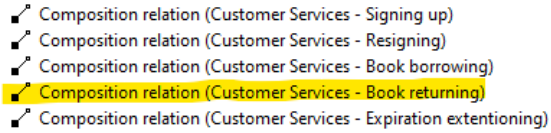


*Figure 12: Internal model on composition of customer services*

3. As described in §2.2.2, ArchiMate can use different graphical notations to represent the same structural relation. Moreover, the right diagram in Figure 6 can be used to represent a Composition or Aggregation relation. Although the correct relation is registered in the internal model after inquiry of the modeler, still misinterpretation of the model can occur by the concerning stakeholder. This could be seen as an inconsistency between the view and the internal model.

4. Another potential inconsistency is related to the richness of the ArchiMate language. ArchiMate has a rich set of elements and relations. Some of these elements and relations are tightly related and even have overlap in the potential use. For example the Goal element is defined as “A goal

represents a high-level statement of intent, direction, or desired end state for an organization and its stakeholders" (AMS, §6.3.1). The Outcome element is defined as "An outcome represents an end result" (AMS, §6.3.2). Those definitions leave room for interpretation and discussion how specific aspects of an enterprise must be classified. Business Processes and Business Functions are also closely related as both describe internal behavior performed by a business role (AMS, §8.3.2). The potential overlap of these elements is also seen by ArchiMate itself: "Although the distinction between the two is not always sharp, it is often useful to distinguish a process view and a function view on behavior" (AMS, §8.3). Using one element in one view and another element in anther view to represent the same aspect of the enterprise, is an inconsistency between views.

## 2.4. Approaches to Achieve Consistency of EM models in other modeling languages

### 2.4.1. Consistency of EM models in 4EM

The modeling language 4EM (Sandkuhl et al., 2014) uses six sub-models as depicted in Figure 13 to model the enterprise. Because the integration of these sub-models adds significantly to the understandability and useability of this model, each sub-model should be connected internally and with other sub-models (Sandkuhl et al., 2014). To archive this integration, Sandkuhl et al. (2014) apply a set of guidelines:

- There should be at least one goal in the Goal Model, one process, one external process, one information/material set in the Business Process Model, one concept in the Concept Model and one actor in the Actors and Resource Model.
- Every Information or Material Set in the Business Process Model must be related to a concept in the Concept Model.
- Every Process must be motivated by at least one goal from Goal Model in some decomposition level
- Every Process must be related to at least one Actors and Resource Model role
- Every information system goal and requirement must be related to at least one goal or business process.

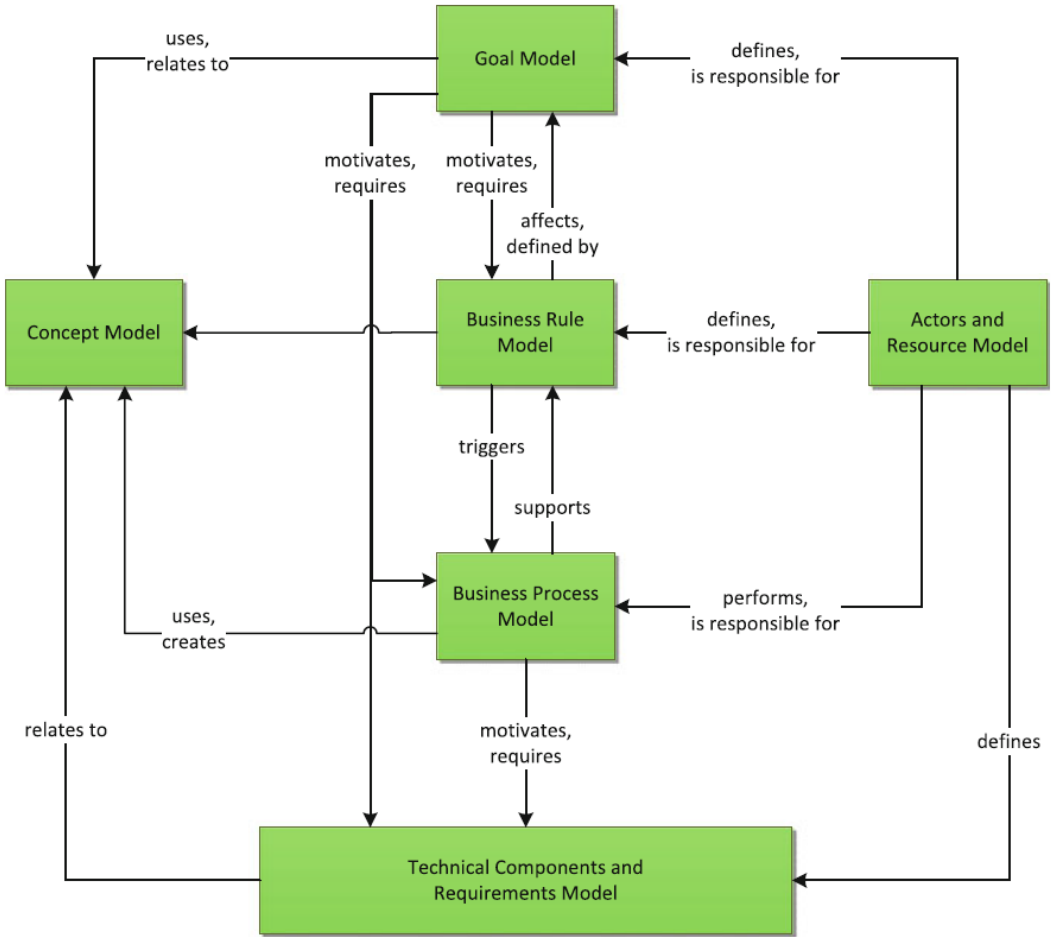


*Figure 13: Sub-models of 4EM; Source: (Sandkuhl et al., 2014)*

Additionally, the modeler should take care that the discussed concepts in the Goal Model should be represented in the *Concept view* and the *Business Process view*. Although the relation is not always obvious and needs some human creativity and interpretation (Sandkuhl et al., 2014).

When comparing 4EM to ArchiMate, a first difference is the fact that 4EM does deliver guidelines for a method of modeling, while ArchiMate does not. Furthermore, 4EM states that each sub-model should be connected internally and with other sub-models, which is also not required by ArchiMate. Where ArchiMate models the core elements of all aspects potentially on all levels in a single model, 4EM has four of its sub-models (Actors and Resource Model, Business Rule Model, Business Process Model and Concept Model) acting on a business level. Only the Technical Components and Requirements Model links to both IT systems and applications, comparable to the application Layer and the Technology Layer of ArchiMate.

### 2.4.2. Consistency of EM models in ExtREME

Within Interactive Modeling and Simulation, Roubtsova (2016) uses the ExtREME method for goal modeling and requirements management as part of a cyclic process to execute system requirements simulation.

Using a goal refinement tree based on GORE concepts, e.g. the KAOS method (Dardenne et al., 1993), requirements are created containing countable and (or) comparable concepts. The requirements are used to derive concepts, of which the behavior is modeled using protocol machines (Roubtsova, 2015). The protocol models, which contain one or more protocol machines,

can structurally be seen as behavioral models (Roubtsova, 2016). Therefore the consistency of ExtREME models is derived from the goal model to both concepts and behavior, modeled together within the protocol model.

When comparing ExtREME to ArchiMate, a first difference is the fact that ExtREME does deliver guidelines for a method of modeling, while ArchiMate does not. Where ArchiMate models the core elements of all aspects potentially on all levels in a single model, ExtREME uses a Goal model and a Protocol model. No concrete separation is used to model on different abstraction levels.

## 2.5. Results and conclusions of the specification and literature review

Review of the ArchiMate Specification and experiments with ArchiMate models shows that inconsistencies can occur in the ArchiMate model during modeling. Comparison with other EM methods shows that inconsistencies of ArchiMate models are caused by the absence of a method that leads the modeler to the choice of abstractions, elements and relations.

# 3. Method design

## 3.1. Method

This research aims at providing a modeling method to create a consistent enterprise model in ArchiMate, based on a case description. As no such method is found during the literature review documented in §2.1, a method has been designed. For this design, the concepts of DSR (Hevner, 2007; Hevner, March, Park, & Ram, 2004) are used. This is because the modeling method that is developed in this research, should be founded both on theoretical knowledge and practical requirements. Also this research aims to deliver both the practical method and additions to the knowledge base. With that, it aligns with the rigor cycle and the relevance cycle that DSR prescribes (Hevner, 2007) (Figure 14). The design cycle of DSR also aligns with the practical approach on design and evaluation in the laboratory testing, that is used with the design of the modeling method.

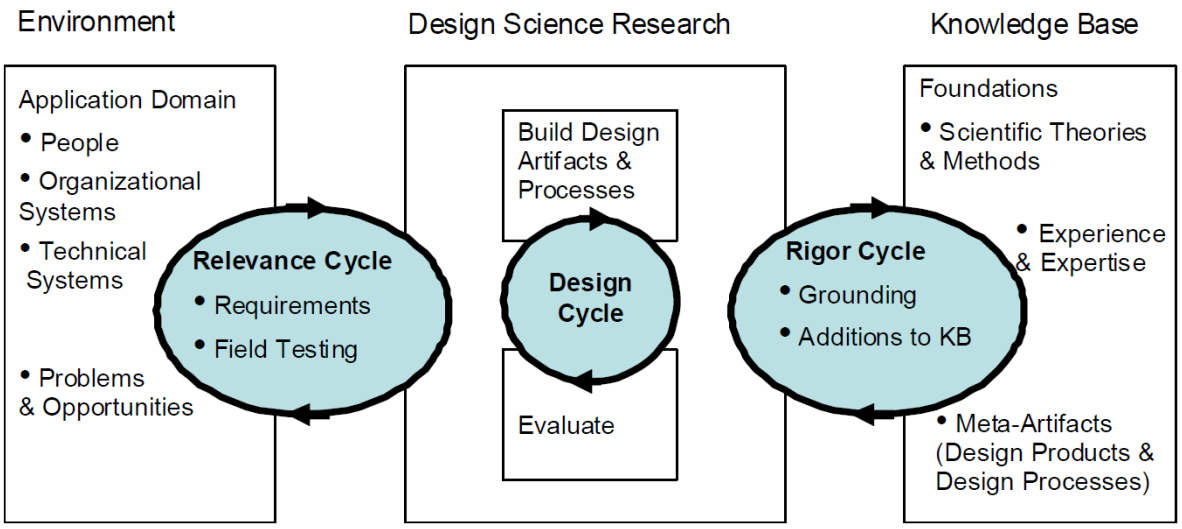


*Figure 14: Design Science Research cycles. Source: (Hevner, 2007)*

With DSR, both theoretical knowledge and practical requirements are used as input for the design cycle. In chapter 2 of this document, a literature research is conducted to obtain this theoretical knowledge, as being part of the Rigor Cycle of DSR. On the one hand the theoretical knowledge about ArchiMate is collected by both reviewing articles on the background of ArchiMate and its current specification. On the other hand, two other EM methods are reviewed on their approach on consistency in enterprise modeling.

The design of the modeling method starts with the specification of the requirements to the method itself, as part of the Relevance Cycle of DSR. The actual Design Cycle takes those requirements together with de collected knowledge, and creates the concrete steps of the modeling method. This design is evaluated with the library example as part of the Design Cycle.

DSR uses multiple iteration strokes on the three cycles to bring the resulting design to a high level of quality. As this research is conducted in a limited timeframe, only one cycle will be executed which might therefore limit the quality of the resulting design.

## 3.2. Requirements for the modeling method

As part of the relevance cycle of DSR, Hevner (2007) states that “Good design science research often begins by identifying and representing opportunities and problems in an actual application environment”. Although there is a main goal in this research to build a modeling method for consistent modeling in ArchiMate, more guidance on requirements for EM Methods is required. In a panel discussion on ‘How to Build a Perfect Enterprise Modeling Method’, Ralyté et al. (2021) list a group of generic quality requirements for an EM method (Table 2).

*Table 2: Generic quality requirements for an EM Method. Source: (Ralyté et al., 2021)*

| | |
|---|---|
| 1. Minimal Redundancy | The constructs used in the different perspectives / layers should have a minimal overlap to avoid redundancy in the models created for the perspective / layer |
| 2. Cross-notational constraints | When one modeling language makes references to constructs used in another modeling language (used for another perspective / layer), then the EM method should contain suitable constraints for using these cross-references |
| 3. Consistency | Changes in one model involving concepts that have cross-notational links to other models should be translated into semantically equivalent changes that an EM modeling tool should automatically apply to all affected models in order to retain an overarching model representation of the enterprise |
| 4. Ontological grounding/ semantics | The constructs should be related to an agreed-upon ontology of artefacts occurring in enterprises, such as resources, processes, events, and so forth. |
| 5. Detection of modeling flaws | The perfect EM method should have advanced tools to detect flaws in models as well as "undesired model patterns". |
| 6. Extensibility | The core constructs of the EM method should be extensible by domain-specific constructs, ideally defined as specializations of the core constructs. The new constructs shall not violate the semantics of the core constructs that they are subclasses of. |
| 7. Multi-level abstractions | Enterprise models describe artefacts at different abstraction levels. |

Note that Ralyté et al. (2021) see an EM Method in the wider context, including modeling language(s), guidance to build the models and potentially a supporting tool. As in our research the language and the tool are fixed to ArchiMate and Archi, respectively, our to-be-designed modeling method aligns with the ‘guidance to build the model’ from Ralyté et al. (2021). For developing the modeling method, the quality requirements as described in Table 3 are used. As the generic requirement ‘Extensibility’ relates primary to the choice of modeling language and modeling tool, it is left out for the development of the modeling method.

*Table 3: Concrete quality requirements for the modeling method*

| | |
|---|---|
| Ontological grounding / semantics | The method should support the main semantics of goal models, concept models and behavior models |
| Minimal redundancy | Use the minimum of three views. |
| Cross-notational constraints | The rules should be defined between goal-concepts, goal-behavior semantics and conceptual-behavior semantics. |

| | |
|---|---|
| Consistency | How to check the cross-notational rules- manually or automatically |
| Detection of modeling flaws | To avoid and detect double elements and unused elements in the internal model of ArchiMate |
| Multi-level abstractions | Create elements on multiple layers within ArchiMate |

Besides the quality requirements, Ralyté et al. (2021) point out that a perfect EM modeling method should take the organizational context into consideration. This theme returns in various forms: Being the capabilities of the specific employees who have to create the model, to the value proposition of the EM method for the organisation. As our method design does not focus on a specific organization, those elements cannot be taken into account. Most of the findings in the conclusion of Ralyté et al. (2021) also relate to the chosen modeling language and tool, or the organizational context. In our method design however, we take into account the 'Strong theoretical foundation', as DSR requires this also.

As situational requirements for our modeling method, the following is defined: It should help filling the internal model of ArchiMate with non-duplicating elements. Besides that, it should result in a model with consistent views.

### 3.2.1. Limitations to the span of the modeling method design

Due to the limited time span and capacity of this research, some limitations to the span of the design are added. To start with, this research is limited to the layers of the core framework of ArchiMate. On top of the core framework, this research does use the Motivation aspect, which includes the Goal and Requirement element. This research limits the definition of the rules to the rules between the goal-concepts and goal-behavior semantics. The rules between the conceptual-behavior semantics will not be covered, although both views will still be consistent with the goal view.

The research uses the most concrete elements of ArchiMate to depict active, passive and behavior elements. E.g. the Business Actor is used for elements executing behavior, as also a Business Role or even the more complex Business Collaboration or Business Interface could be used.

The most widely accepted view on Enterprise Architecture conceptualizes it as the current state (AS-IS), future state (TO-BE) and the transition roadmap (Kotusev, 2019). This research focuses on creating models of the TO-BE state.

## 3.3. Modeling method for consistent modeling in ArchiMate

EM languages use multiple sub-models/ views to model the complete enterprise (Kotusev, 2019; Roubtsova, 2016; Zachman, 1987). In most EM languages, these multiple sub-models/ views are linked together to a complete model of the enterprise by the semantics of the different views and the rules defined between them. Instead, ArchiMate uses a single internal model to model the complete enterprise and link the different views (Lankhorst et al., 2010). This can be compared to the drawings of a house: Other methods use multiple 2d drawings to sketch the complete house, where ArchiMate uses a single 3d drawing. This supports an important part of the consistency of the model: Elements being used in one model/ view are the same elements that are used other models/ views, as long as the elements are linked via the central model. To stay in the equation with the

house: A chimney shown in the 'west' drawing cannot be different or missing in the 'north' drawing when a central 3d model is used.

As the single model in ArchiMate would quickly be getting large during modeling and is only available in a textual description, ArchiMate uses graphical drawings as views on this single model to communicate the model with the stakeholders. Those views are not specifically prescribed and are just a subset of the elements and relations from the internal model. However ArchiMate suggests viewpoints as restriction of element types for the various views. Those viewpoint often combine elements from different layers and/or different aspects of the internal model (AMS; Chapter C).

Although using a single internal model, ArchiMate still divides the model into different aspects. Those aspects are somewhat in line with views within other EM languages. ArchiMate uses the Passive Structure aspect and the Active Structure aspects which are respectively in line with the Concept Model and the Actors and Resource Model of 4EM (Sandkuhl et al., 2014) and the Concept Model of ExtREME, incorporated within the protocol model (Roubtsova, 2016). The Behavior aspect is in line with the Business Process Model of 4EM and the Behavior Model of ExtREME, incorporated within the protocol model. The Motivation aspect of ArchiMate (AMS, §3.5 and §4.4) is in line with the Goal models of both 4EM and ExtREME. Although not specifically defined within the ArchiMate specification (AMS; Chapter C), this research uses the commonly used 'Concept sub-model', 'Behavior sub-model' together with the 'Goal sub-model' as minimum for an Enterprise Model.

In some modeling languages, including ArchiMate, the goal model/ Motivation aspect plays a role in preventing potential inconsistencies in the modeling of the enterprise. Sandkuhl et al. (2014), state in the explanation of the 4EM method that "The Goals Model describes essentially the reason, or motivation, for components in the other sub-models". So as the only reason for existence of elements in other sub-models is an origin in the goal model, the elements in the other views must be derived from or at least related to the goal model. A goal model is also a set of elements (goals, requirements) and refinement relations. The name of each goal is a sentence. The elements of other views can be taken from (lexical) analysis of goal names aimed to identify elements (nouns) and their relations (pairs of nouns labelled with a verb). In order to be useable as a reference for the other models, goals must be refined to countable and (or) comparable concepts (Roubtsova, 2016). Refinement of requirements can be done by splitting in state space, and splitting in time (milestones). With state space decomposition, the subgoals are related to different objects. Using the milestone-approach, a goal is split into several subgoals defining different states in time (Roubtsova, 2016). Refining the goals using both decompositions, results in the active elements, their lifecycles and a logical order of the events.

The specific purpose of the enterprise model plays a role in the level that the Motivation aspect refers to. When goals are defined from an overall business perspective, refinement will take place also on the business level. The derived passive elements, active elements and behavior elements will also be elements on the business layer of the ArchiMate model. Additional, goals can be defined or refined to elements that exist on the application or technology layer. E.g. a constraint to use a specific business to business technical protocol in the communication with customers, can create requirements that directly relate to elements on the lower layers.

Based on the above, a modeling method is proposed to create consistent ArchiMate models as follows:
***Step 1: Refine goals to requirements***
***Step 2: Perform lexical analysis and derive concepts and relations***
***Step 3: Create concept sub-model***

***Step 4: Derive events from requirements***
***Step 5: Create behavioral sub-model***

**Step 1: Refine goals to requirements**

The high level goals of an enterprise should be defined from the case description. Using the concepts of Goal Modeling and Refinement (Roubtsova, 2016), we first start with understanding the assumptions that exist given the domain the enterprise operates in: Those assumptions will not be refined in the Motivation Aspect and are considered self-evident for this domain. For the remaining parts of the goals, one or more concrete assumptions should be extracted from the more abstract goal. Refinement into subgoals can be done by either a state space decomposition or by the milestone approach. The first tactic splits the goal in subgoals that each describe different objects. Most of the time this involves an 'AND' split, where all subgoals together fulfill the concerning goal. But also an 'OR' split is possible, where either of the subgoals can fulfill the concerning goals. The second tactic describes the same objects, but with different states in time. This refinement of a goal creates a tree where all leaves represent requirements (Roubtsova, 2016).

The Goal Refinement Tree can be modeled in ArchiMate by creating one or more views and creating the applicable Motivation elements and their relations in these views. The elements and relations will then be created in the internal model and displayed in the applicable view. ArchiMate uses the Realization relation (AMS, §5.1.4) to refine elements in the Motivation Aspect. By default however, no realization relation is allowed by the same types of motivation elements (AMS, §B.5). The relationships.xml file in the Archi tool can be edited to allow those relations ("Relationships.xml editing,"). This allows for using goal elements as subgoals of other goal elements.

Figure 15 shows the result of step 1 for the library example. Goals are refined to subgoals which end up as requirements.

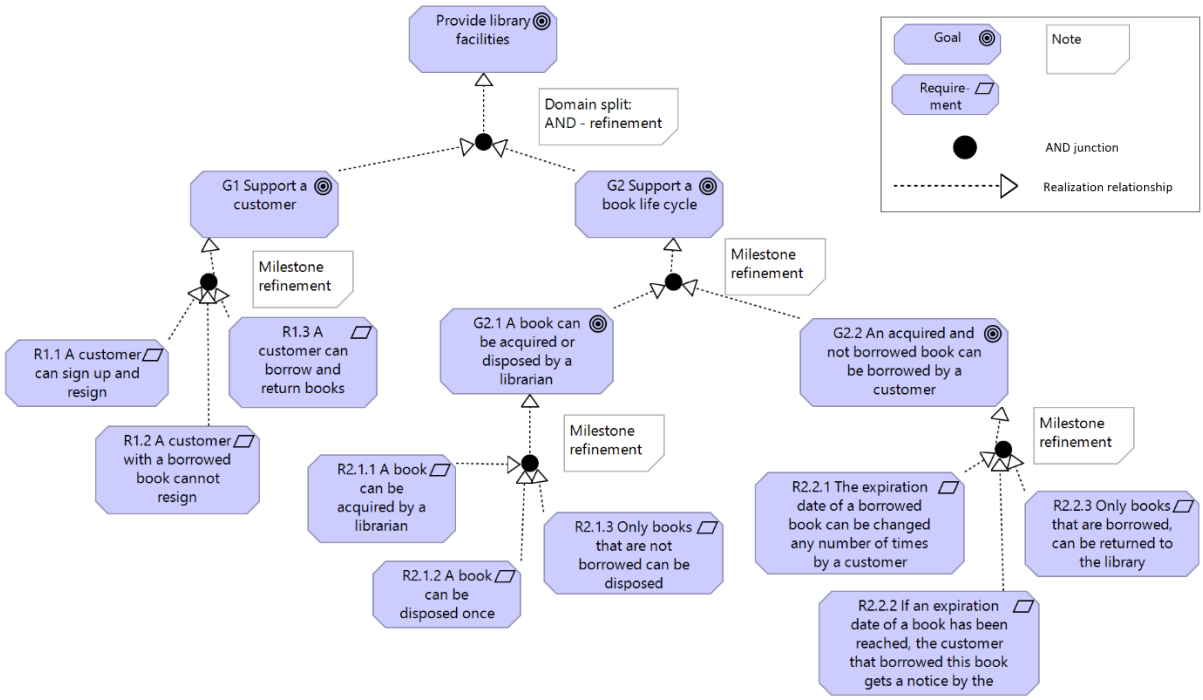


*Figure 15: Motivation aspect of the library example*

**Step 2: Perform lexical analysis and derive concepts and relations**

Goals and requirements are expressed as sentences in an informal spoken language. By lexical analysis of those sentences, elements of the enterprise can be derived (Roubtsova, 2016). Nouns in those sentences are candidates for elements of the passive or active structure, together called concepts. Whether this noun should be seen as active or passive element can be indicated when we parse the sentence rationally: The subject of the sentence can be seen as the acting noun and can be related to an active element. The direct object of the sentence is often the noun that is acted upon, and can be related to a passive element.

The verbs in the sentences indicate relations between the concepts. The relations are specified as a set (verb, subject, object). In Table 4, the concepts and relations are derived from the requirements for the library example.

*Table 4: Derived concepts and relations for the library example*

| **Requirement** | **Concepts** | **Relations** |
|---|---|---|
| R1.1 A customer can sign up and resign | Customer | - |
| R1.2 A customer with a borrowed book cannot resign | Customer<br>Book | (Borrows, Customer, Book) |
| R1.3 A customer can borrow and return books | Customer<br>Book | (Borrows, Customer, Book)<br>(Returns, Customer, Book) |
| R2.1.1 A book can be acquired by a librarian | Librarian<br>Book | (Acquires, Librarian, Book) |
| R2.1.2 A book can be disposed once by a librarian | Librarian<br>Book | (Disposes, Librarian, Book) |
| R2.1.3 Only books that are not borrowed can be disposed | Book | - |
| R2.2.1 The expiration date of a borrowed book can be changed any number of times by a customer | Customer<br>Book | (Changes expiration date, Customer, Book) |
| R2.2.2 If an expiration date of a book has been reached, the customer that borrowed this book gets a notice | Customer<br>Book<br>Librarian | (Notifies, Librarian, Customer) |
| R2.2.3 Only books that are borrowed, can be returned to the library | Book | - |

The core elements must be further specified depending on the layer of the ArchiMate framework on which they reside. E.g. an object related to the business layer can be modeled as a Business Object element, related to the application layer as an Data Object.

Figure 16 shows the graphical representation of step 2 for the library example. In this case, the active elements are created as Business Actors. For the passive element, Business Objects are used in ArchiMate. The relations between the concepts are depicted with a directed Association Relation (AMS §5.2.4).

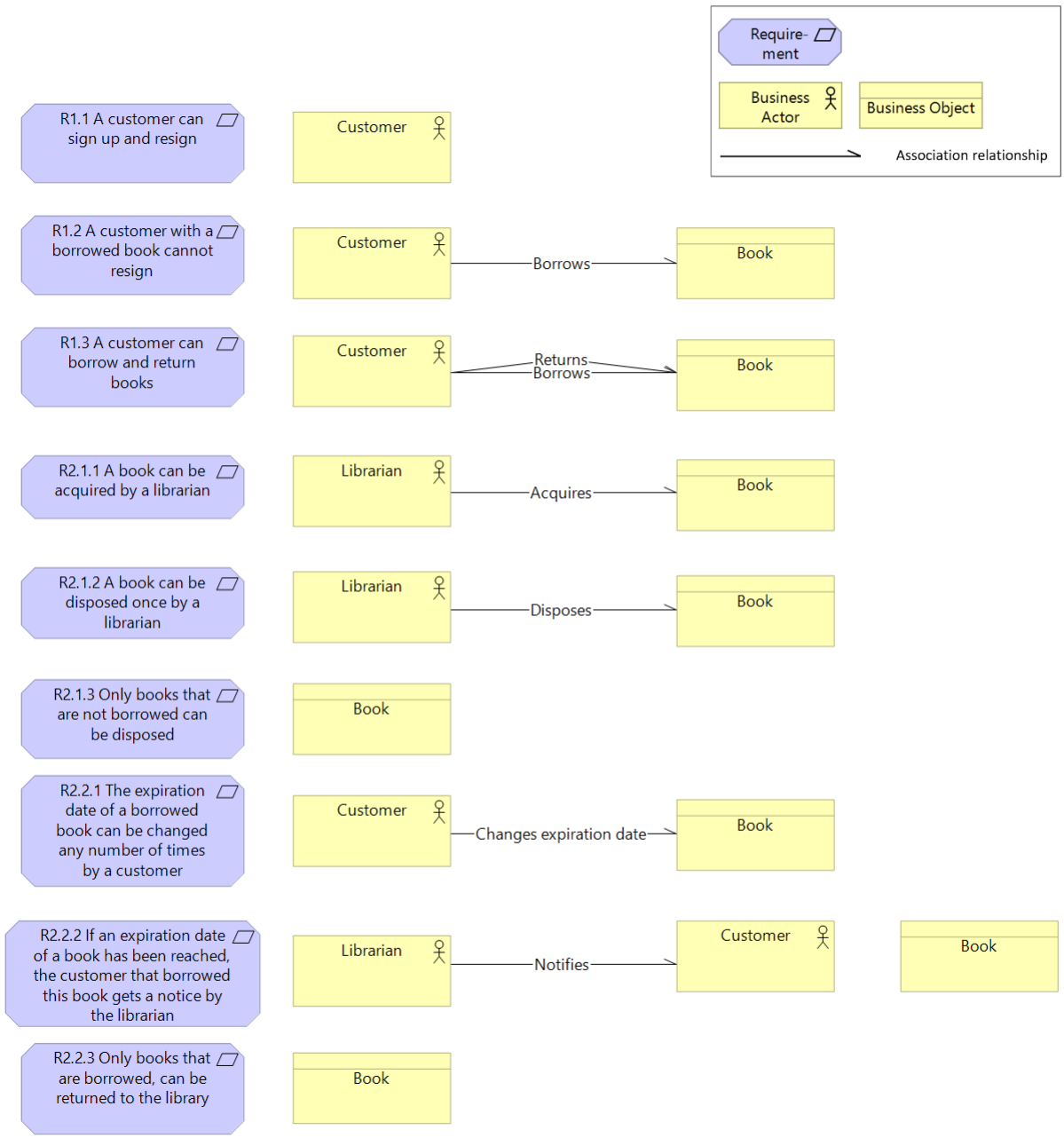


*Figure 16: Graphical representation of the derived concepts and relations for the library example*

**Step 3: Create concept sub-model**

With the elements and relations derived in step 2, a concept sub-model is created.

As the elements and relations are already in the internal model of ArchiMate, this sub-model can be created by creating a new view and dragging the active and passive elements into the view (Figure 17).

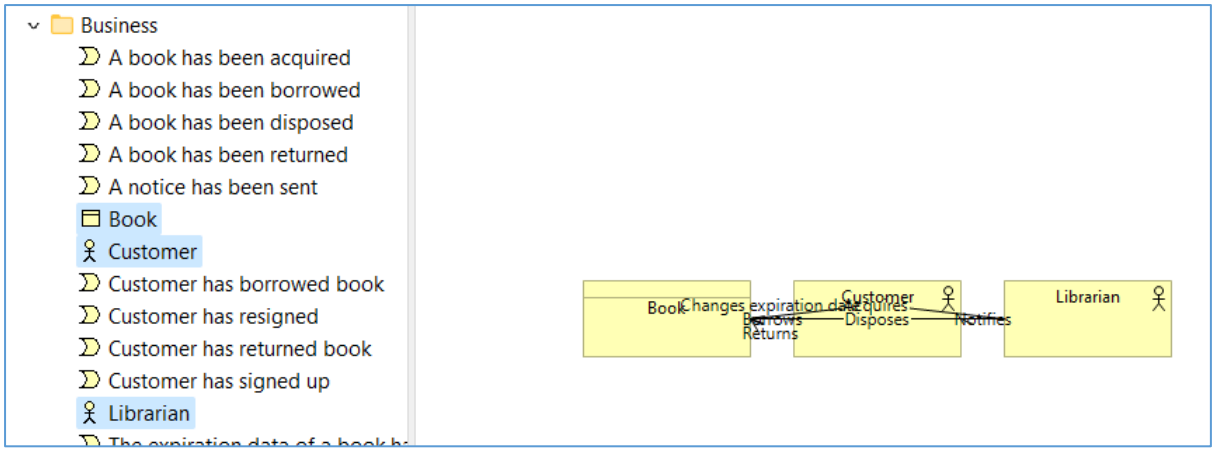


*Figure 17: Creating concept sub-model in Archi using drag&drop*

After visual rearranging the elements and relations, and adding visual grouping on layers and aspects, the concept sub-model for the library example is shown in Figure 18.

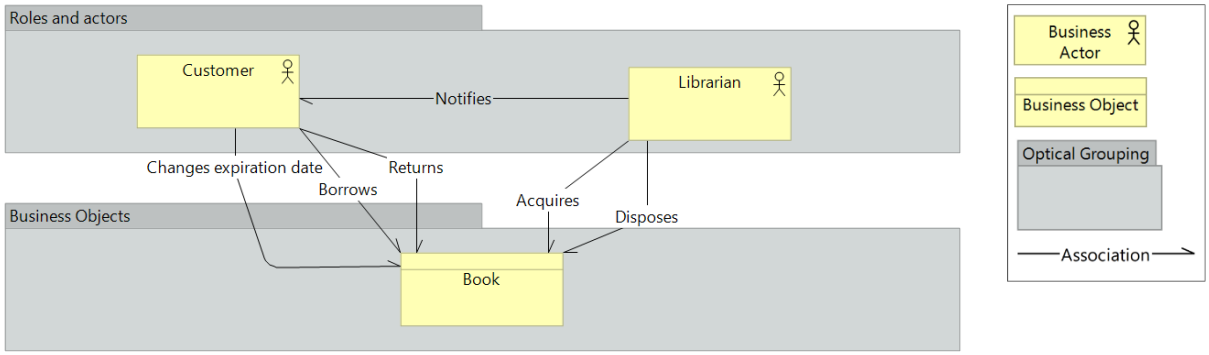


*Figure 18: Concept sub-model for the library example*

**Step 4: Derive events from requirements**

The leaves from a goal model, being the requirements after milestone refinement, represent events for a behavior model (Roubtsova, 2016). From these events, the related concept can be found at the last above domain split in the tree refinement. This concept is used to group the events for that concept. The results for the library example are shown in Table 5.

*Table 5: Derived events and related concepts for the library example*

| **Requirement** | **Events** | **Related Concept** |
|---|---|---|
| R1.1 A customer can sign up and resign | Customer has signed up<br>Customer has resigned | Customer |
| R1.2 A customer with a borrowed book cannot resign | Customer has resigned | Customer |
| R1.3 A customer can borrow and return books | Customer has borrowed book<br>Customer has returned book | Customer |
| R2.1.1 A book can be acquired by a librarian | A book has been acquired | Book |
| R2.1.2 A book can be disposed once by a librarian | A book has been disposed | Book |

| | | |
|---|---|---|
| R2.1.3 Only books that are not borrowed can be disposed | A book has been disposed | Book |
| R2.2.1 The expiration date of a borrowed book can be changed any number of times by a customer | The expiration date of a book has been changed | Customer |
| R2.2.2 If an expiration date of a book has been reached, the customer that borrowed this book gets a notice | A notice has been sent | Customer |
| R2.2.3 Only books that are borrowed, can be returned to the library | A book has been borrowed<br>A book has been returned | Book |

**Step 5: Create behavioral sub-model**

With the events derived in step 4, Event elements are created in ArchiMate. As described in step 2 of this method, this can be a Business Event, Application Event or Technology Event depending on the intentions in the case description. The events are grouped by their related concept to form basic behavioral models for each concept. Between the Events, triggering relations (AMS, §5.3.1) are used to indicate the (potential) order of events occurring. This order must be taken from information in the requirements, but additional information from the case description and/or domain knowledge can be required. Figure 19 shows the behavior sub-model for the library example.

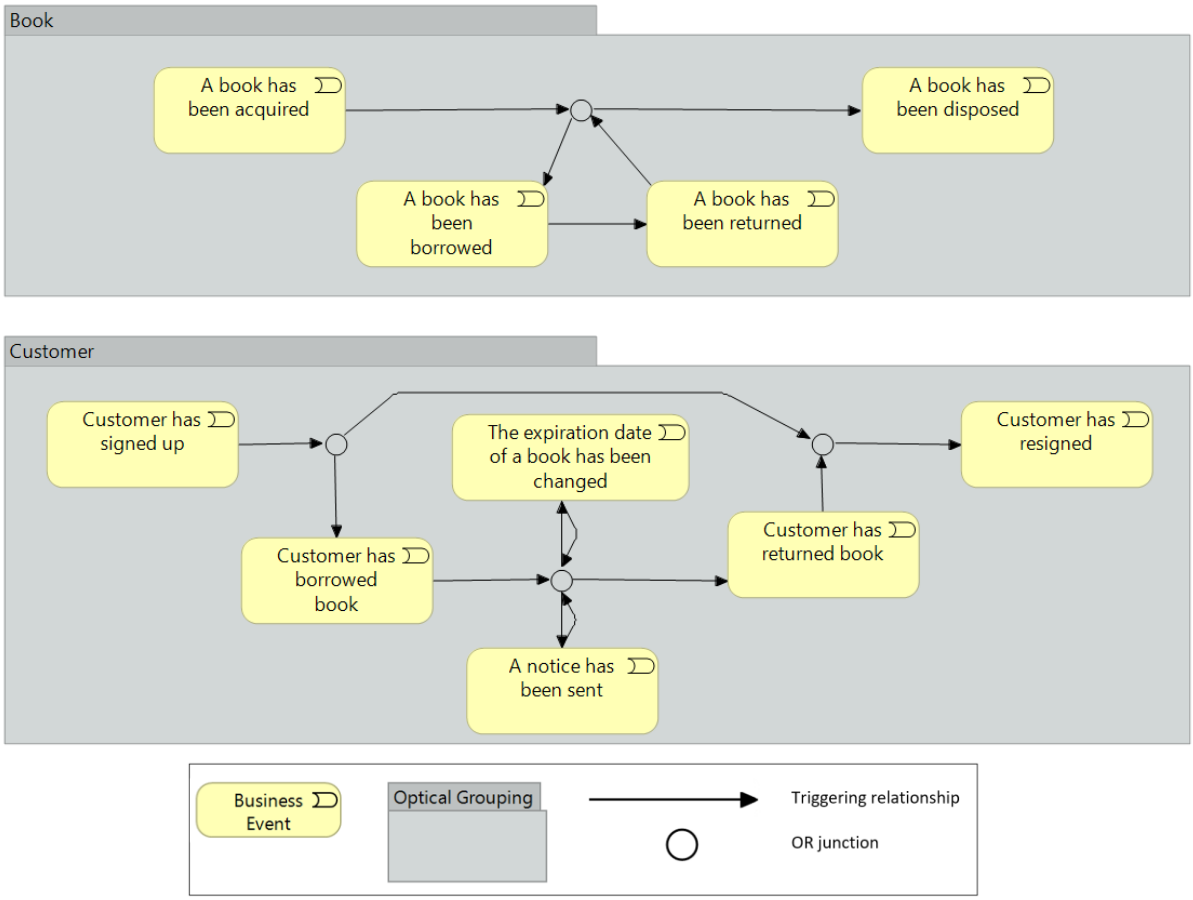


*Figure 19: Behavior sub-model for the library example*

## 3.4. Results on the designed method

Table 6 shows the evaluation of the designed modeling method against the requirements of §3.2.

*Table 6: Evaluation of the requirements for the modeling method*

| Generic requirement | Specific requirement | Result |
|---|---|---|
| Ontological grounding / semantics | The method should support the main semantics of goal models, concept models and behavior models | The method uses the goal sub-model, the concept sub-model and the behavior sub-model. |
| Minimal redundancy | Use the minimum of three views. | The method uses three 'views' on the enterprise, being the goal sub-model, the concept sub-model and the behavior sub-model. |
| Cross-notational constraints | The rules should be defined between goal-concepts, goal-behavior semantics and conceptual-behavior semantics. | The internal model of ArchiMate is used to secure the cross-notational rules for the goal-concepts and goal-behavior semantics. The conceptual-behavior semantics are indirectly secured as both the conceptual model and the behavior model are derived from the goal model. Also the behavior elements are grouped on the main passive elements (in the library example this concerns the book and the customer). |
| Consistency | How to check the cross-notational rules- manually or automatically | There is manual derivation of the concepts, relations and events from the goal sub-model. |
| Detection of modeling flaws | To avoid and detect double elements and unused elements in the internal model of ArchiMate | The method avoids double and unused elements. Detection of double and unused elements is not included in the method as none should be created using the method. Still a check on double and unused elements can be done with the validation tool in Archi. |
| Multi-level abstractions | Create elements on multiple layers within ArchiMate | The method provides in modeling elements on the three core abstraction layers of ArchiMate. |
| Situational requirements | It should help filling the internal model of ArchiMate with non-duplicating elements. Besides that, it should result in a model with consistent views. | The method fills the internal model with non-duplicating elements.<br>Using the elements from the internal model, consistent views are created. |

# 4. Case Study

## 4.1. Design of the case study

As step in the DSR approach, field testing is executed as part of the relevance cycle. For this, a single case study on a non-specific case is used which validates the designed method of consistent modeling in ArchiMate. This case study focusses on the one hand on the useability of the modeling method in practice, on the other hand on potential restrictions of its use. The case study executes the steps of the proposed modeling method on a case and reports the results of each step.

## 4.2. Case selection

This research uses a case description that was created by students of the Open Universiteit during an Enterprise Modeling course. The case describes the changes required at a Dutch wholesaler, to comply with changed regulations on bottle deposits. The original Dutch case description is anonymized and added in Appendix 2, together with the original 4EM models, also anonymized. The case description has been translated from Dutch to English and some small additions have been made to increase the understandability and usability of the case description for this research. To make the ArchiMate models manageable within this report, some parts of the case description have been simplified. This limits the amount of elements in the ArchiMate diagrams. The substantive changes are described.

## 4.3. Reflection w.r.t. validity, reliability and ethical aspects

As this case study has a qualitative nature, the role of the quality indication with validity and reliability is contested (Saunders, Lewis, & Thornhill, 2013). The construct validity can be discussed when a repeatable method is presented. Applying it to two case studies, one on the stage of investigation and another on the stage of testing, covers the DSR cycle. The effectiveness of the modeling method is checked by comparing the 4EM model and the created ArchiMate model.

Internal reliability could have been achieved by using more than one researcher for the research. As this research is to be conducted by a single researcher, no such reliability improvement could be achieved. For external reliability, a comparison will be made to the originally created models in 4EM, as added in Appendix 2. This comparison however has limited value to substantiate the reliability, as a different modeling language and method are used. Also the external reliability is limited as this research uses only a single case to test the proposed modeling method, due to time restrictions. Still the testing with the library case during the design phase and the comparison with the 4EM model add to the external reliability.

No personal information is processed in this research. The case description for the case study was anonymized in respect to the concerning wholesaler.

## 4.4. Case description

***Translated and modified case description:***

The Dutch government is changing the regulations for deposits on PET bottles. The goal of this change is to reduce litter by 70-90%, and to recycle 90% of small plastic bottles. The change in the law goes into effect on July 1, 2021. The case organization must comply with this legislation. For this, changes must be made in some business processes and the systems used. This case focuses on the changes in the Cash & Carry process: The return collection of bags and bottles in the branches of the case organization.

The following important changes apply:

- As of June 1, 2021, there will be a deposit on small PET bottles (<1 liter) for soft drinks and waters. Both large and small PET bottles will receive new barcodes and labels, both on the individual bottle and on the outer packaging. This makes it possible to identify which bottles have a deposit and which do not.
- Companies have an obligation to collect all PET bottles that fall under the national system. Note, also for PET bottles that are not sold by that company.
- Bottles must also be able to be returned by people who are not customers of the case organization and therefore do not have a customer card.
- At present, deposit bottles are identified by shape in the collection machines. In the new situation, it will be done by barcode.
- There will be an app of Statiegeld Nederland which can be used to whether a PET bottle falls under the national deposit system. An employee can use this to manually check a bottle when the collection machine does not work or refuses bottles.

***Limitations and assumptions to the original case description***

- The growing logistics concerning the storage and transport of the bags of bottles are removed from the case description
- The same bags are used by customers to return lager amounts of bottles, as by the case organisation to store the individual returned bottles.
- Bags with bottles returned by customers contain only one type of bottle and are labelled with the concerning barcode and number of bottles.
- Customers without a customer card can only deposit bottles via the manual process, as the employee needs to make a specific invoice in this situation.

## 4.5. Modeling the bottle deposits case

With the method steps as described in §3.3, the bottle deposits case is modeled.

**Step 1: Refine goals to requirements**

Derived from the case description, the central goal of the system is defined as 'Support bottle deposits', which is split with a domain split in the goals to support both deposits of single bottles and deposits of bags with bottles. The former is again split with a domain split into the support for manual return of deposit bottles and automated return of deposit bottles (Figure 20). Note that an 'AND' split is chosen here because the system should support both. As the case description prescribes a RVM to be used and the bottle database should be kept up-to-date, an additional domain split is created to support the return of bottles with the RVM and to support the maintenance of the bottle database. Finally the four subgoals, representing separate domains, are split to requirements using the milestone refinement (Figure 20).

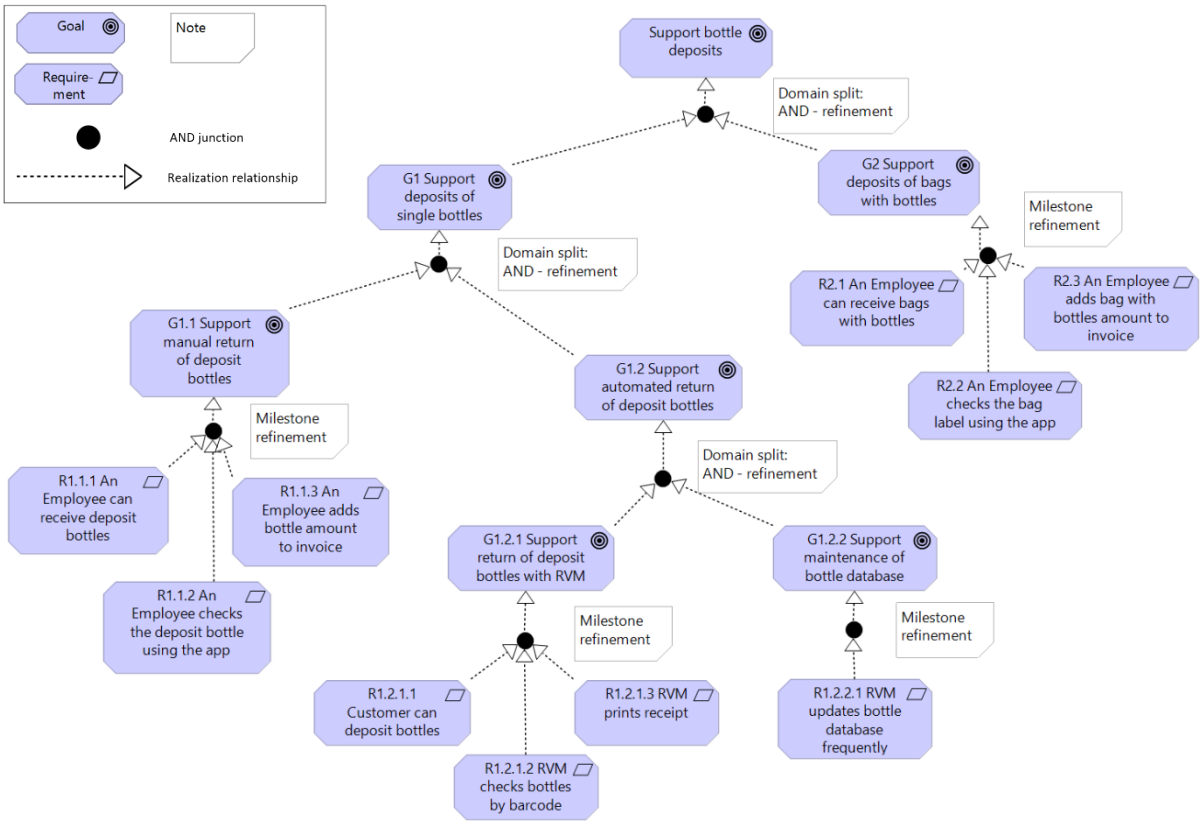


*Figure 20: Motivation aspect of the bottle deposits case*

**Step 2: Perform lexical analysis and derive concepts and relations**

Via lexical analysis the concepts and relations are derived from the requirements, as found in step 1. The derived concepts and relations are listed in Table 7.

*Table 7: Derived components and relations for the bottle deposits case*

| Requirement | Concepts | Relations |
|---|---|---|
| R1.1.1 An Employee can receive deposit bottles | Employee<br>Deposit bottle | Receives |
| R1.1.2 An Employee checks the deposit bottle using the app | Employee<br>Deposit bottle<br>Bottle check app | Checks<br><br>Uses |
| R1.1.3 An Employee adds bottle amount to invoice | Employee<br>Invoice<br>Deposit bottle | Adds amount |
| R1.2.1.1 Customer can deposit bottles | Customer<br>Deposit bottle | Deposits |
| R1.2.1.2 RVM checks bottles by barcode | RVM<br>Deposit bottle | Check by barcode |
| R1.2.1.3 RVM prints receipt | RVM<br>Receipt | Prints |
| R1.2.2.1 RVM updates bottle database frequently | RVM<br>Bottle database | Updates |
| R2.1 An Employee can receive bags with bottles | Employee<br>Bag with bottles | Receives |

| | | |
|---|---|---|
| R2.2 An Employee checks the bag label using the app | Employee<br>Bag with bottles | Checks label |
| R2.3 An Employee adds bag with bottles amount to invoice | Employee<br>Invoice<br>Bag with bottles | Adds amount |

Figure 21 shows the graphical representation of deriving the concepts and relations. In this figure the concepts are shown yellow for elements on the business layer of ArchiMate. Business Actors are created for elements of the active structure, and Business Objects for elements of the passive structure. Some elements represent elements of the application layer (shown in blue), which are prescribed in the case description. For that layer, Application Components are created for elements of the active structure, and Data Objects for elements of the passive structure.

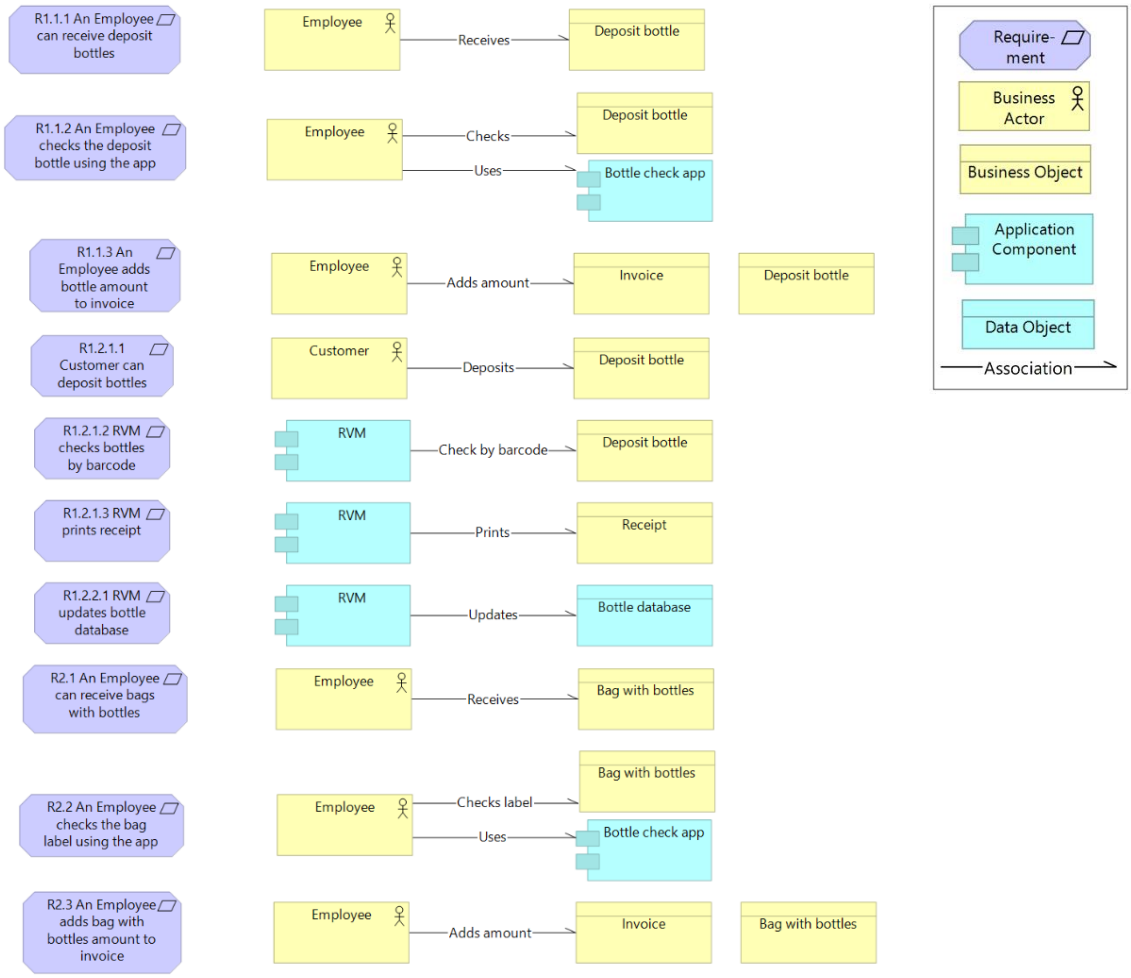


*Figure 21: Graphical representation of the derived concepts and relations for the bottle deposits case*

**Step 3: Create concept sub-model**

A concept sub-model is created by dragging the active and passive elements into the view and rearranging the elements by ArchiMate layers and aspects (Figure 22).

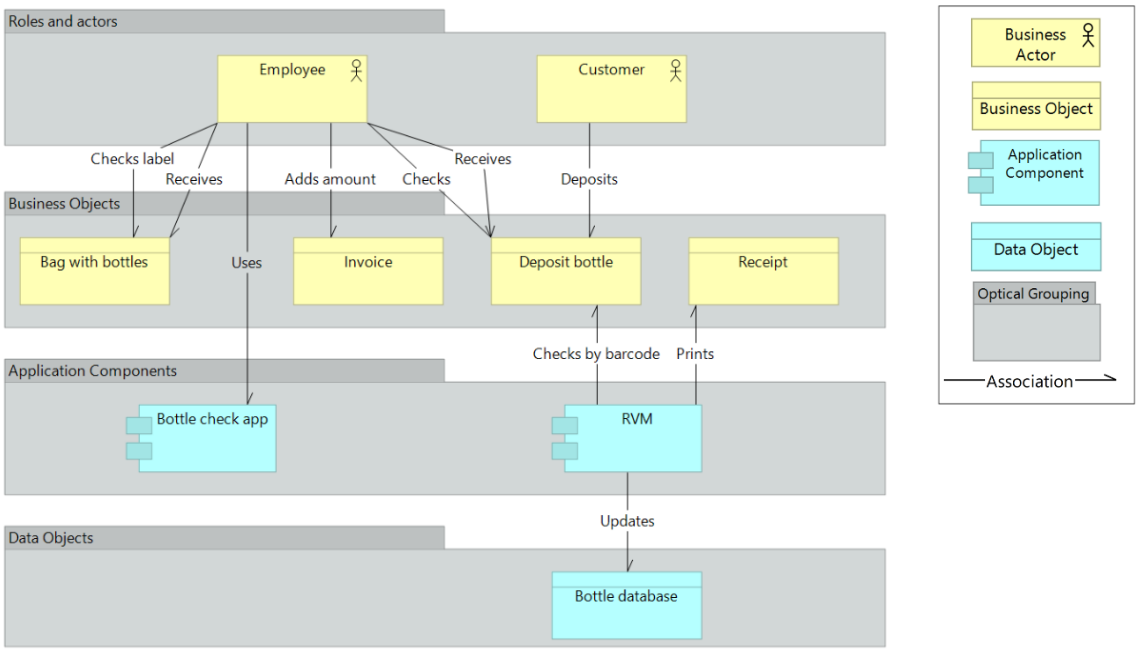


*Figure 22: Concept sub-model for the bottle deposits case*

**Step 4: Derive events from requirements**

The events and related concepts are derived from the goal sub-model (Table 8).

*Table 8: Derived events and related concepts for the bottle deposits case*

| Requirement | Events | Related Concept |
|---|---|---|
| R1.1.1 An Employee can receive deposit bottles | Deposit bottle received | Deposit bottle (manual) |
| R1.1.2 An Employee checks the deposit bottle using the app | Deposit bottle checked | Deposit bottle (manual) |
| R1.1.3 An Employee adds bottle amount to invoice | Amount added to invoice | Deposit bottle (manual) |
| R1.2.1.1 Customer can deposit bottles | Bottle deposited | Deposit bottle (automated) |
| R1.2.1.2 RVM checks bottles by barcode | Bottle checked by barcode | Deposit bottle (automated) |
| R1.2.1.3 RVM prints receipt | Receipt printed | Deposit bottle (automated) |
| R1.2.2.1 RVM updates bottle database frequently | Bottle database updated | Bottle database |
| R2.1 An Employee can receive bags with bottles | Bag with bottles received | Bag with bottles |
| R2.2 An Employee checks the bag label using the app | Bag label checked | Bag with bottles |
| R2.3 An Employee adds bag with bottles amount to invoice | Amount added to invoice | Bag with bottles |

**Step 5: Create behavioral sub-model**

With the events derived in step 4, Event-elements are created in ArchiMate and grouped by the related concept (Figure 23). For the events related to actions of the RVM, Application Events (blue)

are created. The other events are depicted as Business Events as they are related to actions of Business Actors. Triggering relations are added to indicate the (potential) order of events occurring.

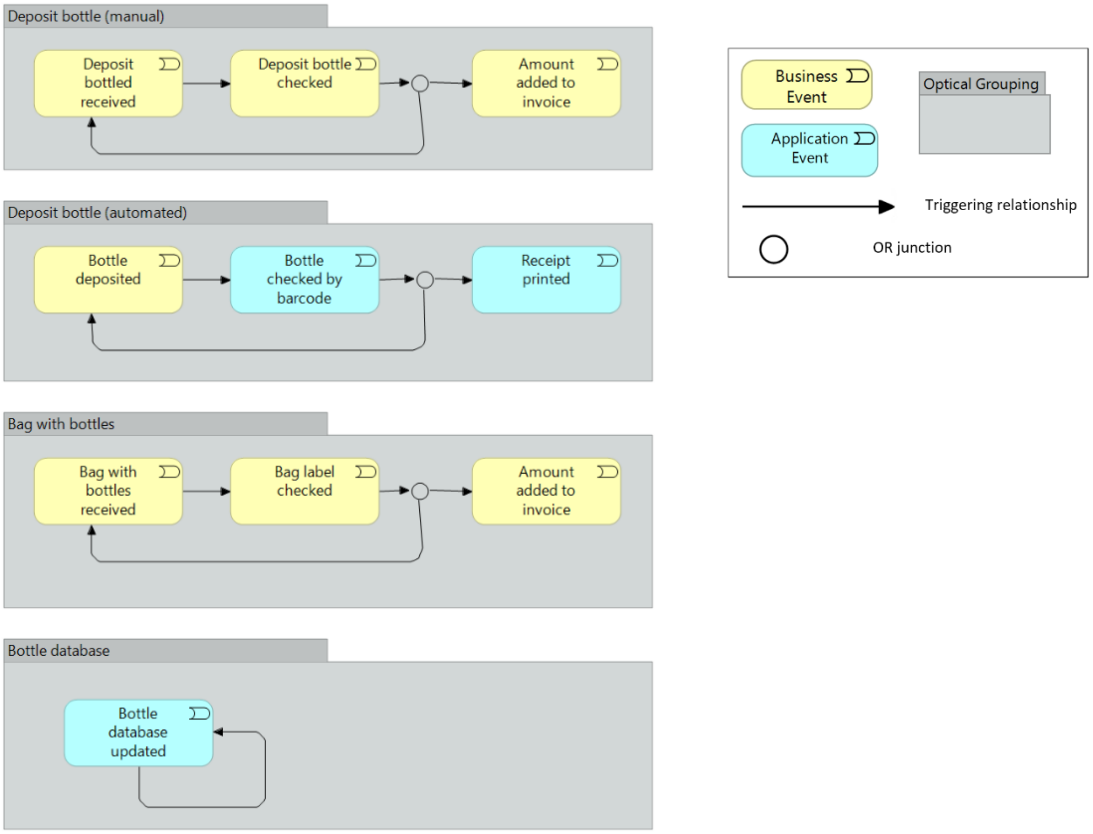


*Figure 23: Behavior sub-model of the bottle deposits case*

## 4.6. Comparing results 4EM model to ArchiMate model

In Appendix 2 the original diagrams of the bottle deposits case in 4EM are included. When comparing those results to the ArchiMate model, the goal (sub-)model shows comparable items. This excludes of course some elements that were on purpose left out in the case modeling within ArchiMate. The 4EM goal model has however less concrete structure on milestone refinements around a single concept.

Comparing the concept (sub-)models in ArchiMate and 4EM gives a similar picture as with the goal model: Comparable concepts are depicted, besides the concepts that were left out on purpose. The ArchiMate model however shows elements on both the business layer and the application layer. In the 4EM model, the additional 'Technical Components and Requirements Model' is used as an application/ technology layer. Remarkable is the difference in the position of the RVM as an application component within ArchiMate but as a (business) concept in 4EM.

The Customer, Employee, RVM and Cash Register ('kassa') are depicted both in the Actors and Resource Model and the Concept Model of 4EM. Their relations in both views differ however, which is an inconsistency between the views. The internal ArchiMate model prevents such an inconsistency when the modeling is conducted by the described method.

Finally, the behavior models of the 4EM model and the ArchiMate model are very different as 4EM uses processes and information sets, where this ArchiMate model uses diagrams based on state machines and events.

## 4.7. Results of the modeling of the bottle deposits case

Using the proposed modeling method in ArchiMate on the bottle deposits case, creates an enterprise model consisting of a goal sub-model, a concept sub-model and a behavior sub-model. As the used core elements and relations are derived from the requirements refined from the goal, a consistent model is created. A final check on the created model using the validator tool in Archi (Figure 24) indicates no double names or unused elements in the internal model.


*Figure 24: Result of the validator tool in Archi on the bottle deposits model*

Comparing the results of the proposed modeling method and the original diagrams from the 4EM model on the deposit bottles case, show the modeling method in ArchiMate can be used to find inconsistencies in models created in other modeling methods.

# 5. Discussion, conclusion and recommendations

## 5.1. Discussion and conclusion

The first research question refers to the basic knowledge on enterprise modeling ArchiMate and consistency within its models. The literature and specification analysis found that ArchiMate was intended to align different views in an enterprise architecture, not necessarily modeled within ArchiMate. The internal model that ArchiMate uses plays a key role in the consistency between the various sub-models. Still some potential inconsistencies in ArchiMate models are identified. Those inconsistencies might occur when modeling is not performed in a systematic way. No such modeling method is defined within the ArchiMate specification of found during the literature analysis.

The second research question refers to modeling methods used in other modeling languages, that safeguard the consistency of the models. Both ExtREME and 4EM use a goal sub-model to derive elements and relations (e.g. concepts and events) for the other sub-models. This links the complete enterprise model in a consistent way. Using their concepts of enterprise modeling and using additional quality requirements from Ralyté et al. (2021), a modeling method is proposed to create consistent enterprise models in ArchiMate. This modeling method fulfills all the specified requirements, except the rules between the conceptual-behavior semantics, which it only partly described in the method.

Laboratory testing of the modeling method, as part of the design cycle of DSR, using the library example shows useability to create consistent enterprise models in ArchiMate, where the same elements and relations in the internal model are used in the different sub-models. The filling of the internal model is driven from the goal sub-model by lexical analysis of the requirements. The requirements on their turn are derived from goals and sub-goals by domain splits and milestone refinement.

Field testing, as part of the rigor cycle of DSR, by a case study using the deposit bottles case, shows that the method can be used on a practical case. This application shows no limitations for its use in practice. This conclusion can however not be generalized for all enterprise modeling activities, due to the fact that only one non-specific case is tested. This also aligns with the concept of DSR that multiple iteration cycles are required to create a solid design. This answers the third research question on the useability of the modeling method in practice.

The conclusion for this research is that the proposed modeling method in ArchiMate can be used to finitely direct the model to the state of testable correspondence with the case description.

## 5.2. Recommendations for further research

The proposed modeling method focusses on the creation of a ‘TO-BE’ model of an organisation. For that situation it is likely that the goals and requirements are indeed established and can bring the model to a consistent state. For an AS-IS model whoever, the current state of an organization is not always the result of intended design based on specified goals (Dumas et al., 2018). Therefore deriving the concepts, relations and events from goals and requirements might not be a realistic option. Additional research should be done on how to create consistent AS-IS models in EM and eventually create a complete set of an AS-IS model, a TO-BE model and the transition roadmap. For the transition roadmap, potentially elements of the Implementation and Migration Layer of ArchiMate could be used as shortly discussed in §2.2.3.

The proposed modeling method is based on concepts of other modeling languages and uses as such the sub-models that are derived from those modeling languages, and also commonly used with enterprise modeling. These sub-models however seem not fully compatible with the modeling intentions of ArchiMate. As example, the modeling of behavior in ArchiMate has only a superficial level, as discussed in §2.2.4. Also the concept sub-model, as used in the method, might not be the intended way to model those relations in ArchiMate. The division of the internal ArchiMate model into 'aspects' might somewhat resemble to the different views used in other modeling languages. E.g. the Actors and Resource Model of 4EM has resemblance with the Active Structure Aspect and the Business Process Model in 4EM has resemblance with the Behavior Aspect of ArchiMate. Finally the Concept Model of 4EM had resemblance with the Passive Structure Aspect of ArchiMate. Additional research could be done on approaching ArchiMate in another way: To integrate detailed models of other modeling languages (Lankhorst, 2004). In such situation, the whole core model of ArchiMate could potentially be seen as a 'concept model', being the central link to the other detailed models. The integration between the ArchiMate model and other models require a specific set of transformations (Lankhorst, 2004). As Lankhorst (2004) states: "The semantic soundness of such transformations is particularly nontrivial and thus requires further exploration". However, Lankhorst (2004) also predicts that "True round-trip modeling between different languages will not always be possible". It might however be feasible to define "semantic checks that point out possible inconsistencies between models at different abstraction levels" (Lankhorst, 2004).

In this research only the most concrete types of elements are derived from the motivation model. E.g. the Business Actor is used for elements executing behavior, as also a Business Role or even the more complex Business Collaboration or Business Interface could be used. Further research could focus on also deriving more complex elements and structural relations between the elements from the requirements. Elements and relations that cannot be derived in a direct manner from the requirements, could potentially be added in additional analytic steps in the modeling method.

# References


Archi. (2021). Archi modelling toolkit. Retrieved from https://www.archimatetool.com/

Archi User Guide. (2021). Retrieved from https://www.archimatetool.com/

ArchiMate® 3.1 Specification. (2019). Retrieved from https://pubs.opengroup.org/architecture/archimate3-doc/toc.html

Belzer, J., Holzman, A. G., & Kent, A. (1975). Encyclopedia of Computer Science and Technology. (25), 73.

Dardenne, A., Van Lamsweerde, A., & Fickas, S. (1993). Goal-directed requirements acquisition. *Science of computer programming, 20*(1-2), 3-50.

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). *Fundamentals of business process management (second edition)*: Springer.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems, 19*(2), 4.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 75-105.

Kotusev, S. (2019). Enterprise architecture and enterprise architecture artifacts: Questioning the old concept in light of new findings. *Journal of Information technology, 34*(2), 102-128.

Lankhorst, M. M. (2004). Enterprise architecture modelling—the issue of integration. *Advanced Engineering Informatics, 18*(4), 205-216.

Lankhorst, M. M., Proper, H. A., & Jonkers, H. (2010). The anatomy of the archimate language. *International Journal of Information System Modeling and Design (IJISMD), 1*(1), 1-32.

OU Library - Advanced search. Retrieved from https://openuniversiteit-summon-serialssolutions-com.ezproxy.elib10.ub.unimaas.nl/#!/advanced

Ralyté, J., Bork, D., Jeusfeld, M. A., Kirikova, M., & Stirna, J. (2021). *Panel Discussion: How to Build a Perfect Enterprise Modeling Method.* Paper presented at the Forum at Practice of Enterprise Modeling 2021 (PoEM-Forum 2021), Riga, Latvia, November 24-26, 2021.

Relationships.xml editing. Retrieved from https://github.com/archimatetool/archi/issues/243

Roubtsova, E. (2015). Advances in behavior modeling. In *Advances in Computers* (Vol. 97, pp. 49-109): Elsevier.

Roubtsova, E. (2016). *Interactive Modeling and Simulation in Business System Design*: Springer.

Roubtsova, E., van Gool, L. C., Kuiper, R., & Jonkers, H. (2002). Consistent specification of interface suites in UML. *Software and Systems Modeling, 1*(2), 98-112.

Sandkuhl, K., Stirna, J., Persson, A., & Wißotzki, M. (2014). *Enterprise modeling - Tackling Business Challenges with the 4EM Method*: Springer.

Saunders, M., Lewis, P., & Thornhill, A. (2013). *Research methods for business students, Seventh edition*: Pearson education.

Zachman, J. A. (1987). A framework for information systems architecture. *IBM systems journal, 26*(3), 276-292.

# Appendix 1: Case description library example

***Case description of library example, copied from (Roubtsova, 2016). Pages 89-90.***

A library system provides the basic functionality for registering customers and handling books.

First of all, there are customers who interact with a library: sign up, borrow a book, and resign. There are also books being the library's resources. A new book can be acquired and disposed by a librarian. An acquired and available book can be borrowed. If a book is borrowed, the expiration date of the lending can be changed any number of times. Only books, that are borrowed, can be returned to the library. Only books that are not borrowed can be disposed. A book can be disposed once. No further actions with a disposed book are possible. Books can only be borrowed if the customer is signed up, i.e., he is an active customer of the library. The signed customer can resign only once. A resigned customer cannot perform any further actions.

There are two additional business rules in our library:

• A customer with a borrowed book cannot resign. Only customers without "any debts" are able to cancel their membership.

• If an expiration date of a book has been reached, the customer borrowed this book gets a notice.

And finally, we recognize two roles with two actors.

• The role "customer" represents a member of the library. She(he) can sign up, resign, borrow and return books.

• The role "librarian" represents someone working in the library. A librarian can acquire new books, dispose them and send a warning notice to a member who have not returned a book on time.

# Appendix 2: Original Dutch case description and 4EM models

***Original Dutch case description of the bottle deposits case, anonymized***

Er is een wettelijke wijziging vanuit de overheid voor het statiegeldproces voor PET-flessen, waar een Nederlandse groothandel aan moet voldoen. De wettelijke wijziging heeft tot doel het zwerfaval met 70 tot 90% te verminderen en 90% van de kleine plastic flesjes te recyclen. Deze wettelijke wijziging gaat in zijn volledigheid in per 1 juli 2021.
Er is dus noodzaak gevonden in het ontwikkelen van verbeteringen in het bedrijfsproces.
De huidige systemen en processen van de groothandel sluiten namelijk nog niet aan op dit nieuwe wettelijke statiegeldproces. De wijzigingen hebben onder andere impact op het Food Service Delivery (FSD) proces, het Cash & Carry proces, de retourzendingen en financiële afhandeling met Statiegeld Nederland.

Dit probleem is van grote omvang. Vandaar dat wij ons gaan focussen op het probleem binnen het Cash & Carry proces: de retour-inname van zakken en flessen in de vestigingen van de groothandel.
De volgende grote wijzigingen zijn van toepassing:
Per 1 juni komt er statiegeld op kleine PET-flessen (<1 liter) voor frisdranken en waters. Zowel grote als kleine PET flessen krijgen nieuwe barcodes en etiketten, zowel op de losse fles als op omverpakkingen. Er vindt dan controle plaats of het flesje wel/geen statiegeld oplevert.
De groothandel heeft een verplichting tot inname van alle PET flessen die onder het landelijk systeem vallen voor grote en kleine PET flessen voor frisdranken en waters. Let wel, ook voor PET flessen die de groothandel niet verkoopt.
Flessen aannemen die van niet klanten van de groothandel zijn, en dus geen klantenpas hebben, moet per 1 juni kunnen.
Volumes van lege PET flessen gaan toenemen in de keten. Dit leidt mogelijk tot wijzigingen in het logistieke proces waar een deel van de ingenomen emballage waarschijnlijk via een centraal magazijn naar Statiegeld Nederland gaat.
Op dit moment worden statiegeldflessen geïdentificeerd aan de hand van vorm in de gebruikte inname machines. Straks gebeurt dat op barcode.
Er komt een app van Statiegeld Nederland waarmee gecontroleerd kan worden of een PET fles onder het landelijk statiegeld systeem valt. Een medewerker kan hiermee een fles handmatig controleren wanneer de innamemachine niet werkt of flessen weigert.

***Original Dutch 4EM models of the bottle deposits case, anonymized***

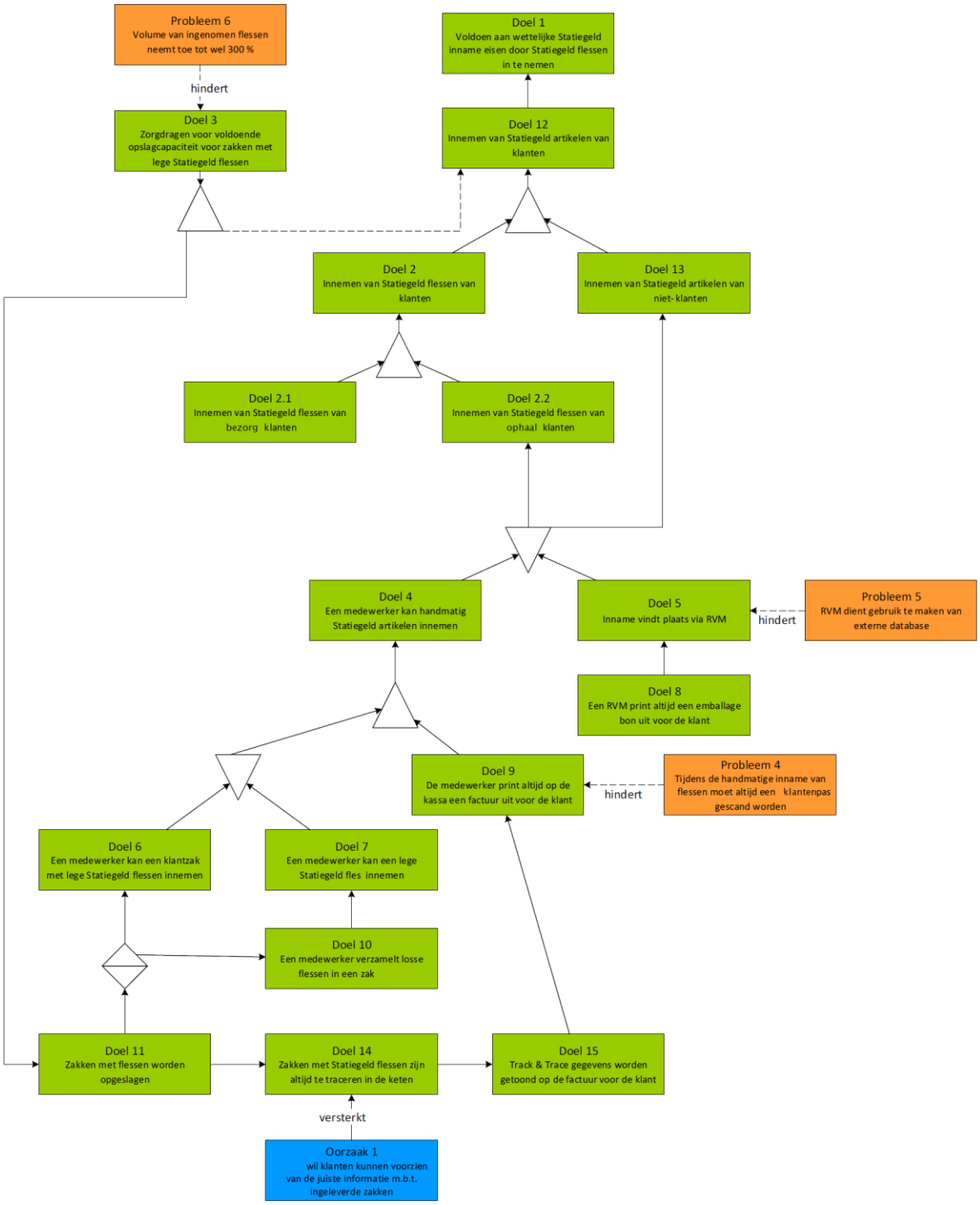


*Figure 25: 4EM Goal model of the bottle deposits case*

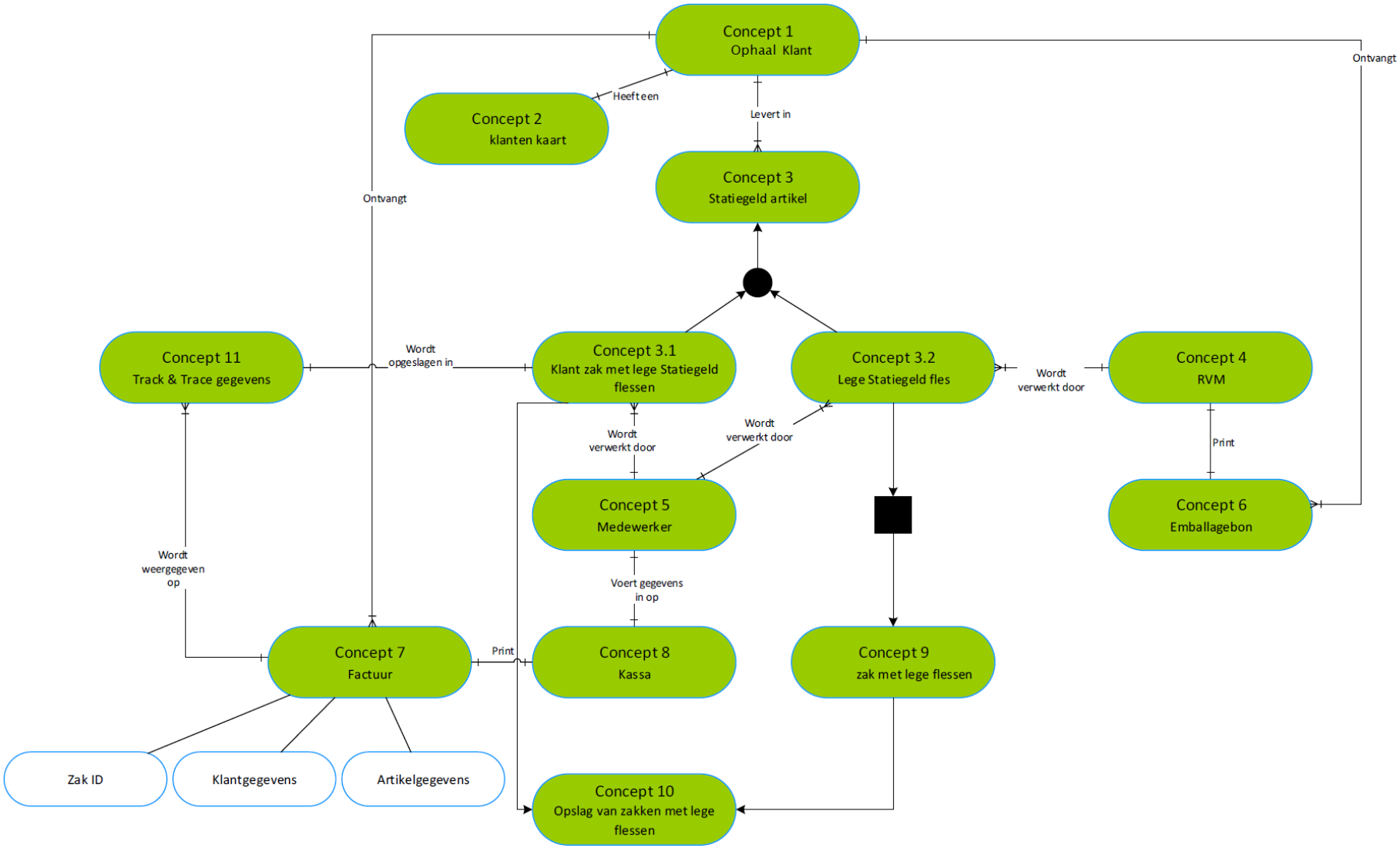


*Figure 26: 4EM Concept Model of the bottle deposits case*

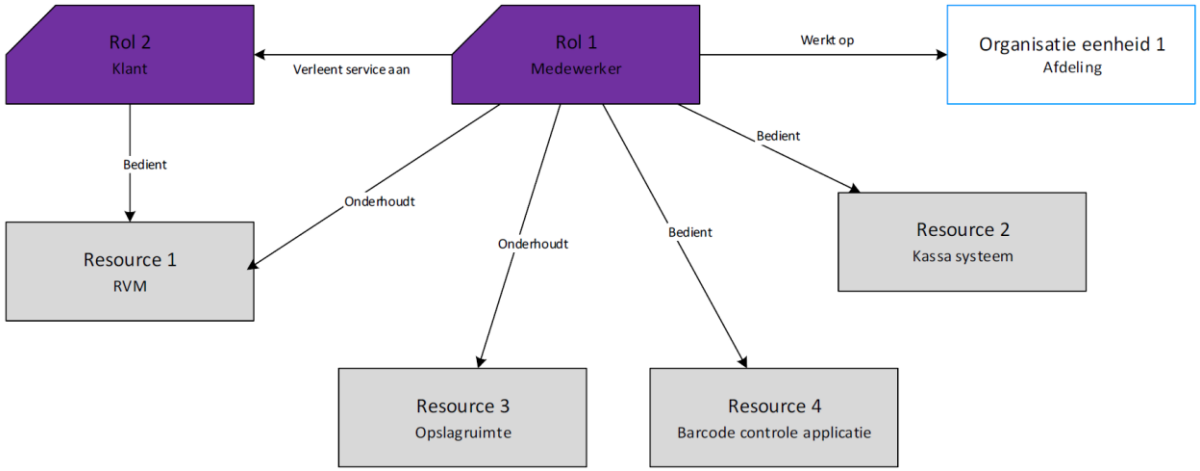


*Figure 27: 4EM Actors and Resource Model of the bottle deposits case*

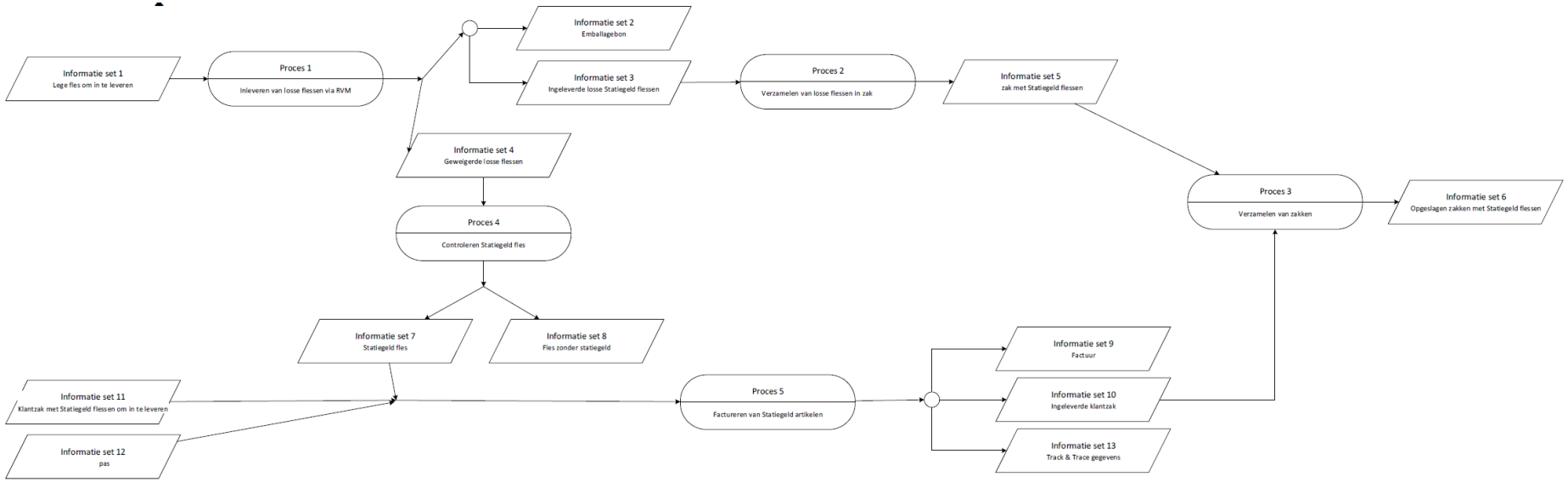


*Figure 28: 4EM Business Process Model of the bottle deposits case*

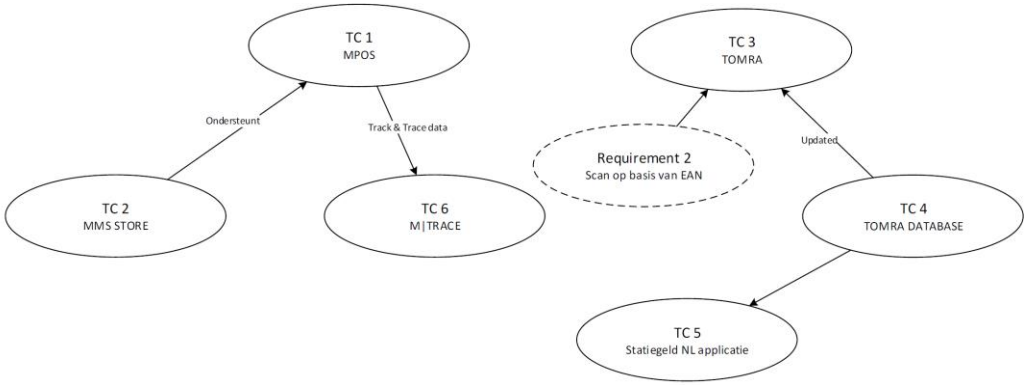


*Figure 29: 4EM Technical Components and Requirements Model of the bottle deposits case*