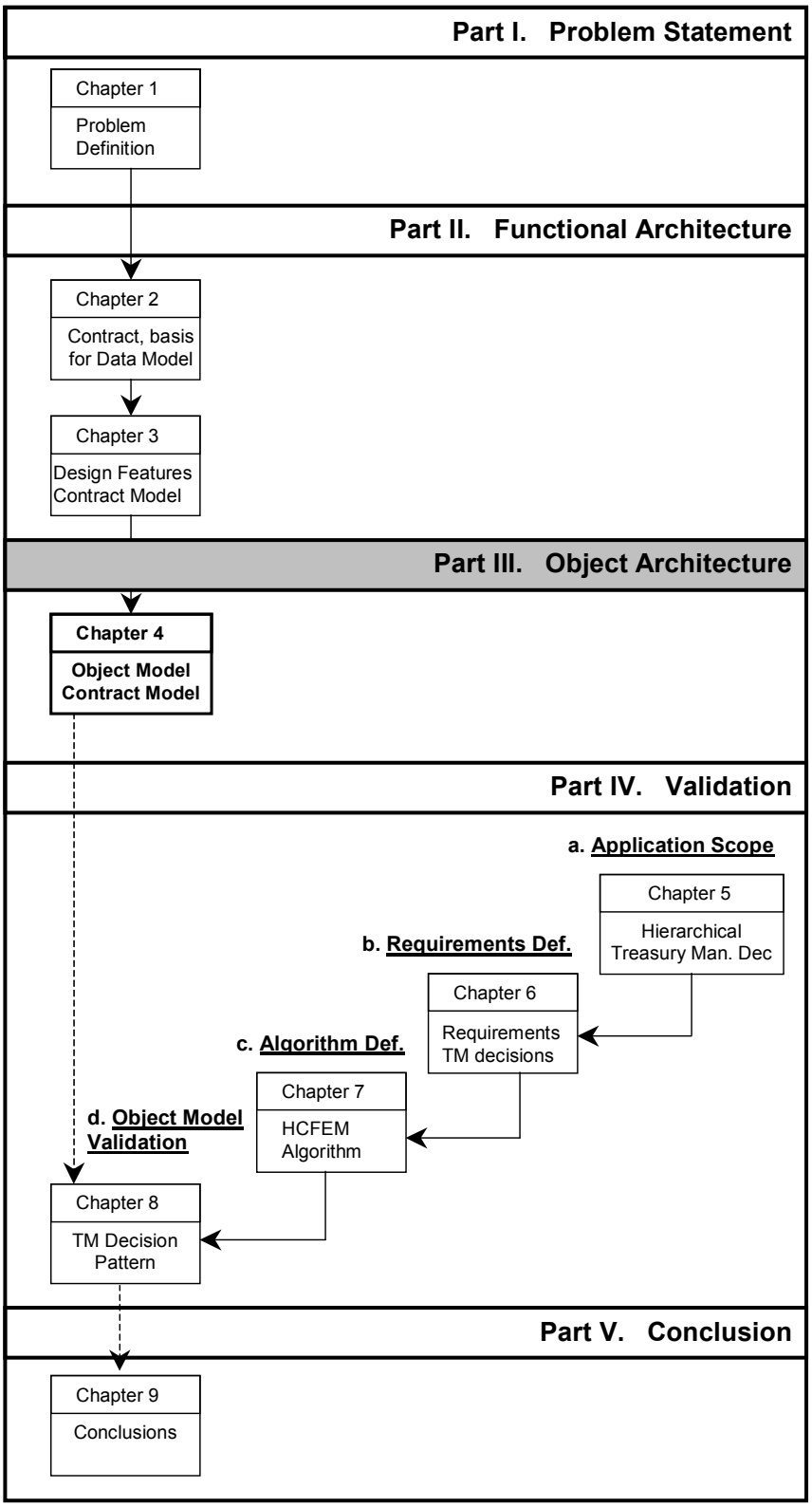
Accounting information for changing business needs

Vandenbossche, P.E.A.

Part III: Object Architecture


Part I. Problem Statement
Chapter 1
Problem Definition
Part II. Functional Architecture
Chapter 2
Contract, basis for Data Model
Chapter 3
Design Features Contract Model
Part III. Object Architecture
Chapter 4
Object Model Contract Model
Part IV. Validation
a. Application Scope
Chapter 5
Hierarchical Treasury Man. Dec
b. Requirements Def.
Chapter 6
Requirements TM decisions
c. Algorithm Def.
Chapter 7
HCFEM Algorithm
d. Object Model Validation
Chapter 8
TM Decision Pattern
Part V. Conclusion
Chapter 9
Conclusions

4. Static Object Model of the Contract Data Model

4.1 Introduction

Part three of this research consists of a technical architecture description to answer research objective one, outlined in Chapter 1, Section 1.4.

> ***Research Objective 1:*** To propose a data organization framework enabling the recording of *ex post* and *ex ante* accounting data on business process instances suitable for supporting *changing* accounting information requests defined by *different* users[23].

The technical architecture of the contract data model involves the modelling of the functional architecture, as proposed in Chapters 2 and 3, as an accounting data model expressed in a generally accepted design modelling language that enables implementation in large business information systems and deployment in real-life customer implementations. This chapter provides an answer to research question three of research objective one as outlined in Section 1.4 of Chapter 1.

> ***Research Question 3:*** How can design features of essential BPI data components be modelled into an accounting data model which can be implemented in large business information systems?

Chapter 4 is structured as follows. An object-oriented analysis model of the definition of resource exchanges is presented first (i.e. the contract clause model) in Section 4.2. Section 4.3 then presents an object analysis model of the execution of resource exchanges (i.e. the fulfilment model). Finally, Section 4.4 presents an analysis model of the full contract data model. This model integrates the requirements of the contract clause model and the fulfilment model. Three different aspects are discussed per model. First, the functional scope of the model is elaborated and then the extent to which the requirements described above are met is described. Lastly, guidance is provided on the implementation choices made in the design process of the class model. All object-oriented design models are expressed in UML[24].

4.2 Contract Clause Model[25]

Functionality
The Contract Clause Model enables the definition of the core objects of the contract data model and is visualized in Figure 4-1. Design Feature 1 states that a contract is a grouping of contract clauses. In other words, the content of a contract is defined by one or more 'contract clauses'. A contract clause contains the details on a resource exchange between participants. An exchange is defined as '*an action whereby the entity foregoes control over some resources in order to obtain control over other resources*' (Ijiri 1975, p. 61).

---

[23] 'Business process instance' is abbreviated as 'BPI' in the remainder of this chapter.
[24] UML (Unified Modelling Language) notation standards are described in Fowler and Scott (1997) and in Appendix 2.
[25] Please refer to Section 3.2.6 of Chapter 3 for Design Features 1 to 9; and to Section 3.3.3 for Design Features 10 to 12.

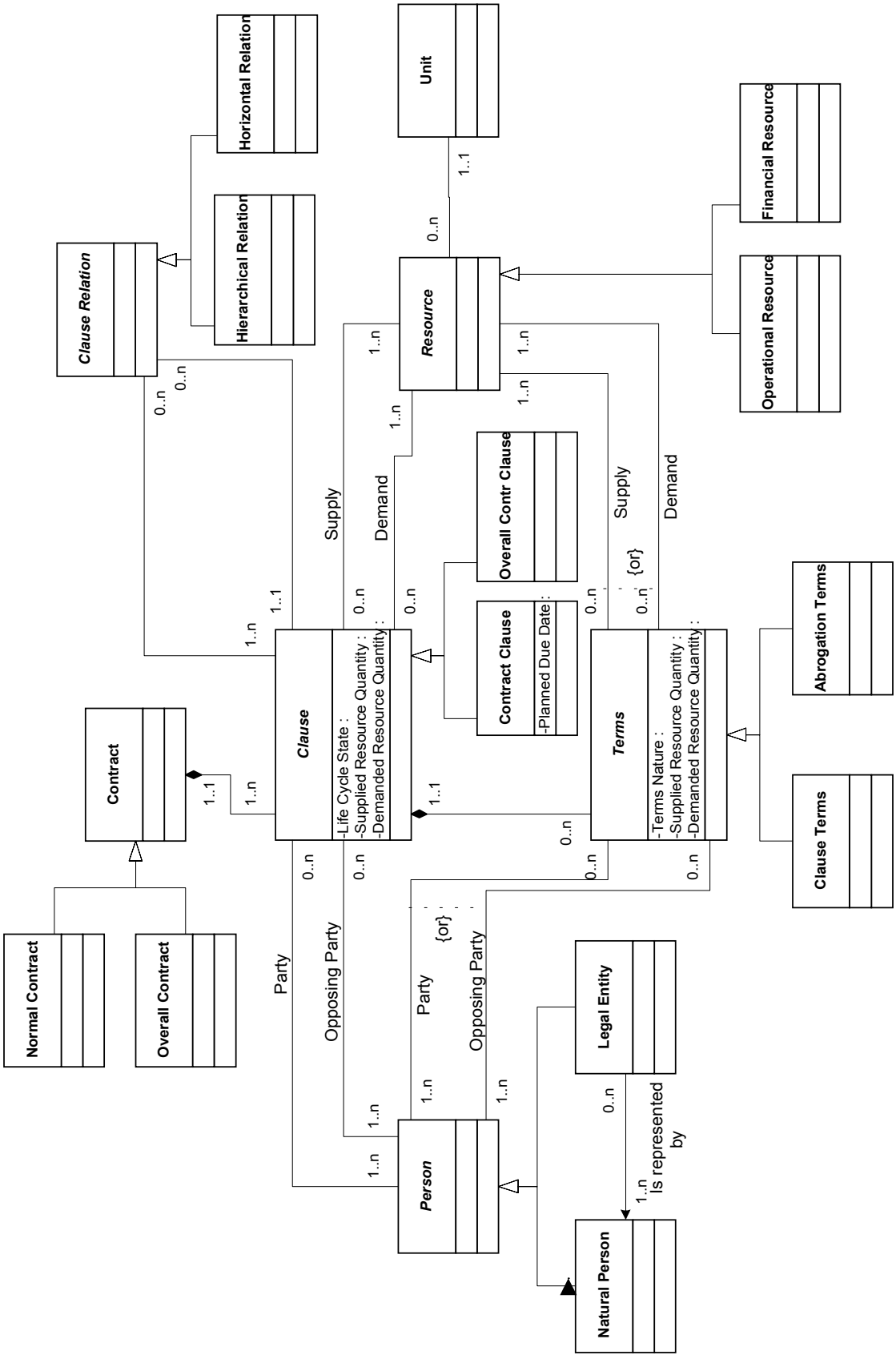


Figure 4-1. Contract clause model

The first relationship specifies the subject of the exchange (the resource *supplied*). In a sales transaction, this is the operational resource being sold, whereas in a purchase transaction it is the operational resource being bought. The second relationship concerns the resource used to compensate for the subject of the contract (the resource *demanded*). Normally, this resource is a specific type of 'financial resource' (e.g. money). The exchange of resources is expressed by means of the [Clause] and [Resource] classes, and their associated specific interrelationships – associations supporting the 'supply' and 'demand' relationships as described earlier in this section. Depending on the resource type, the resources used as subjects or as compensation are defined as belonging to either the [Operational Resource] or the [Financial Resource] class. Both are subclasses of the [Resource] super-class. The quantity of the resource being supplied or demanded is defined by the following properties: 'Supplied Resource Quantity' and 'Demanded Resource Quantity' in the [Clause] class. Quantity is expressed as a unit, implemented by a relationship (association) between the [Resource] and [Unit] classes. Design Feature 10 details that two types of contracts can be discerned. 'Normal contracts' are contracts detailing commitments made by an organization (e.g. an exchange of 'wheelchairs' for 'money'). 'Overall contracts' are contracts recording the conditions of a possible future resource exchange (e.g. the agreed price for the following year for resource exchanges of 'wheelchairs'). The difference between these two types of contracts is that (normal) contracts have an explicit exchange due date, which is not the case in overall contracts. This difference has to be defined at a contract 'clause' level to comply with Design Feature 1. In other words, there are two types of 'clause', the 'contract clause' and the 'overall contract clause'. This is implemented by defining two subclasses for the [Clause] class, the [Contract Clause] subclass that stores normal contract exchanges and the [Overall Contract Clause] subclass that stores exchanges of overall contracts. The due date, which can only be defined for (normal) contract clauses, is implemented by defining the 'Planned Due Date' property in the [Contract Clause] subclass. Other behaviour is the same in both types of contracts (e.g. the definition of a resource exchange, the definition of contract participants, etc.). Design Feature 2 specifies that clause participants can perform two roles, that of 'clause party' or 'clause opposing party'. Section 3.2.2 of Chapter 3 explained that these participants are 'persons' in legal theory. Participants used in the definition of 'clauses' can either be organizations ('legal entities' in legal terminology) or natural persons. These terms are also used in the model. Depending on their type, participants are defined in the [Natural Person] class or in the [Legal Entity] class, both of which are subtypes of the [Person] class. The roles these participants play in the clause are defined using specific interrelationships (associations supporting the 'party' and 'opposing party' roles) defined between the [Person] class and the [Clause] class. The conditions under which the exchange of resources takes place are detailed in 'terms'. As explained in Design Feature 4, two types of 'terms' can be distinguished, 'clause terms' and 'abrogation terms'. 'Clause terms' detail the circumstances under which the execution of the clause takes place. 'Abrogation terms' determine the procedures that have to be followed if one of the clause participants does not execute the contract as described in the clause terms. Depending on type, terms are defined in the [Clause Terms] class or the [Abrogation Terms] class, both of which are subtypes of the [Terms] class. Terms (of both types) are defined as 'rights' or 'obligations' of participants in the 'party' or 'opposing party' role (Design Feature 5). This is implemented by the definition of the 'Terms Nature' property (values: 'right', 'obligation') in the [Terms] class. A 'term' is always defined from the perspective of a participant or participants in one of either role defined for clause (i.e. the 'party' or 'opposing party' roles). For example, the delivery terms detail when participants in one of either role (e.g. the 'party' role) are obliged to deliver. The fact that participants defined in the opposing role (in this case, in the 'opposing party' role) have the right of receipt is not detailed. This is implemented through a specific interrelationship (the 'party' association or the 'opposing party' association, depending on the situation) between the [Person] class and the [Terms] class[26]. Depending on type, a 'term'

[26] The previous paragraph explained that two types of 'Persons' exist for clause participants: natural persons and legal entities. This distinction is also applicable to participants, which are defined at term level.

details either the execution or the abrogation of a resource that is 'supplied' or 'demanded' from the perspective of a selected clause role (i.e. either 'party' or 'opposing party'). This is implemented through one of the specific interrelationships (the 'supply' or 'demand' association, depending on what was chosen in the terms) between the [Resource] class and the [Terms] class. The quantity of the resource, detailed in 'terms', is defined as possessing either the 'Supplied Resource Quantity' or 'Demanded Resource Quantity' property of the [Terms] class. Quantities are expressed in units. This is implemented by a relationship (association) between the [Resource] and [Unit] classes. Design Feature 5 describes clauses passing through different states during their lifecycles (i.e. the 'planned' and 'committed' phases). This functionality is supported by the 'lifecycle state' property, defined in the [Clause] class. Section 3.3 of Chapter 3 noted that contracts are not stored independently of each other. Instead, their definitions were found to be highly structured, comparable to a portfolio where hierarchical and horizontal relationships relate contracts. In keeping with Design Feature 1, the object model should not support a contract portfolio in which relationships are defined at contract level but rather a contract clause portfolio where relationships are defined at contract clause level. The clause portfolio consists of hierarchical relationships between clauses (Design Features 10 and 11) and horizontal relationships between clauses (Design Features 10 and 12). Relationships between clauses are implemented through associations between the [Clause] class and the [Clause Relationship] class. Depending on their type, relationships are stored in the [Hierarchical Relationship] class or the [Horizontal Relationship] class, both classes subtypes of the [Clause Relationship] class.

Requirements

The following requirements are supported by features offered in the Contract Clause Model.

Requirement 1. Data on a single BPI to be defined coherently. In the model proposed, data used in a single BPI are limited to the data on resource exchanges. Exchanges are defined in the [Clause] class in relation to the supply and demand relationships (associations) with resources, defined in the [Resource] class. As explained in Design Feature 1, a contract can contain several contract clauses, so data on different resource exchanges that belong together are defined coherently. The Contract Clause Model does not support the coherence of data on a contract's fulfilments. This functionality is supported in the Fulfilment Model (see Section 4.3).

Requirement 2. Data defined in BPIs is ex ante *data* (the second part of this requirement, that BPI data is *ex post* data, is supported in the Fulfilment Model, see Section 4.3). BPI data, defined in contracts and clauses, are defined over various phases of their lifecycle. 'Planned' relates to *ex ante* data (see Design Feature 9). This is implemented by the 'lifecycle state' property, defined in the [Clause] class.

Requirement 3. Data between different BPIs to be defined coherently. The Contract Clause Model holds the data needed to support the requirements of the 'contract portfolio'. Two different types of relationship can be defined between clauses, hierarchical relationships and horizontal relationships (Design Features 11 and 12). Hierarchical relationships are defined in the [Hierarchical Relationship] class and horizontal relationships are defined in the [Horizontal Relationship] class.

Implementation Choices

Two types of clauses can be distinguished (see Design Feature 10). These two types are defined in the [Contract Clause] class and in the [Overall Contract Clause] class. These two classes are subclasses of the [Clause] super-class. A 'contract' (defined in the [Contract] class) is a grouping of several clauses (defined in the [Clause] class). Depending on the type of clauses that are grouped, it is also possible to distinguish between two types of contracts. These two types are defined in the [Normal Contract] and the [Overall Contract] classes (see

Design Feature 10). These two classes are subclasses of the [Contract] super-class. Natural persons (defined in the [Natural Person] class) and legal entities (defined in the [Legal Entity] class) can have exactly the same roles in 'clauses' as in 'terms'. They can play the roles of 'party' and 'opposing party' (see Design Feature 3), implemented through the association relationships. Therefore, an abstract [Person] class holding the common behaviour and properties is introduced to optimize the design. The [Person] class is divided into the [Natural Person] and the [Legal Entity] sub-classes. The association relationships that are common to both subclasses are now defined for the abstract [Person] class. This is illustrated in Figure 4-1. Operational resources (defined in the [Operational Resource] class) and financial resources (defined in the [Financial Resource] class) can perform exactly the same roles in clauses and terms. They can play a 'supply' or 'demand' role in an exchange definition (i.e. as clauses), or in a definition of circumstances of an exchange (i.e. as terms). Therefore, an abstract [Resource] class holding the common behaviour and properties is introduced to optimize the design. The [Resource] class has the [Operational Resource] and the [Financial Resource] classes as subtypes. The association relationships that are common to both subclasses are now defined for the abstract [Resource] class. This is illustrated in Figure 4-1. Clause terms (defined in the [Clause Terms] class) and abrogation terms (defined in the [Abrogation Terms] class) have exactly the same behaviour and properties. Both are defined as either a 'right' or an 'obligation' of participants in the role of 'party' and 'opposing party' on one or more resources. Therefore, an abstract [Terms] class holding the behaviour and properties common to both is introduced to optimize the design. The [Terms] class is divided into the following subclasses, the [Clause Terms] class and the [Abrogation Terms] class (see Design Feature 4). The following common behaviour is defined for the abstract [Terms] class – the 'Terms Nature' property (values: 'right' and 'obligation', see Design Feature 5), the 'supply' and 'demand' specific association relationships, defined in the [Resource] class, and the 'party' and 'opposing party' specific association relationships, defined in the [Person] class. This is illustrated in Figure 4-1. In the object model Figure 4-1, it was decided to define participants as playing the roles of 'party' or 'opposing party' only in contract clauses and not in contracts, as the approach proposed can already cope with the most complex scenario envisaged, where participants are different per clause of the contract, though admittedly, participants will usually be the same in all contact clauses. In the latter event, this should imply that specific interrelationships (associations supporting the 'party' and 'opposing party' roles) can be defined between the [Person] class and the [Contract] class.

4.3 Fulfilment Model[27]

Functionality

The functionality supported by the Fulfilment Model concerns the support of contract clause fulfilments and is visualized in Figure 4-2. Design Feature 8 introduces fulfilments to define and store information on how clauses are executed. Design Feature 1 explained that a contract is only a grouping of contract clauses. Likewise, a fulfilment is only a grouping of fulfilment lines. Therefore, information on 'fulfilments' is defined in one or more fulfilment lines. Because the fulfilment line holds the information on the execution of a contract clause, it is expected that the functionality of a fulfilment line is comparable to the functionality of a contract clause. The functionality of a fulfilment line is therefore described in comparison with the functionality of a contract clause. A contract clause contains the detail on the exchange of resources between participants. It was explained in Section 4.2 that some resources make up the *subject* of the clause whereas other resources comprise what is offered as *compensation* for the resources defined in the clause subject.

[27] Please refer to Section 3.2.6 of Chapter 3 for Design Features 1 to 9, and to Section 3.3.3 for Design Features 10 to 12.

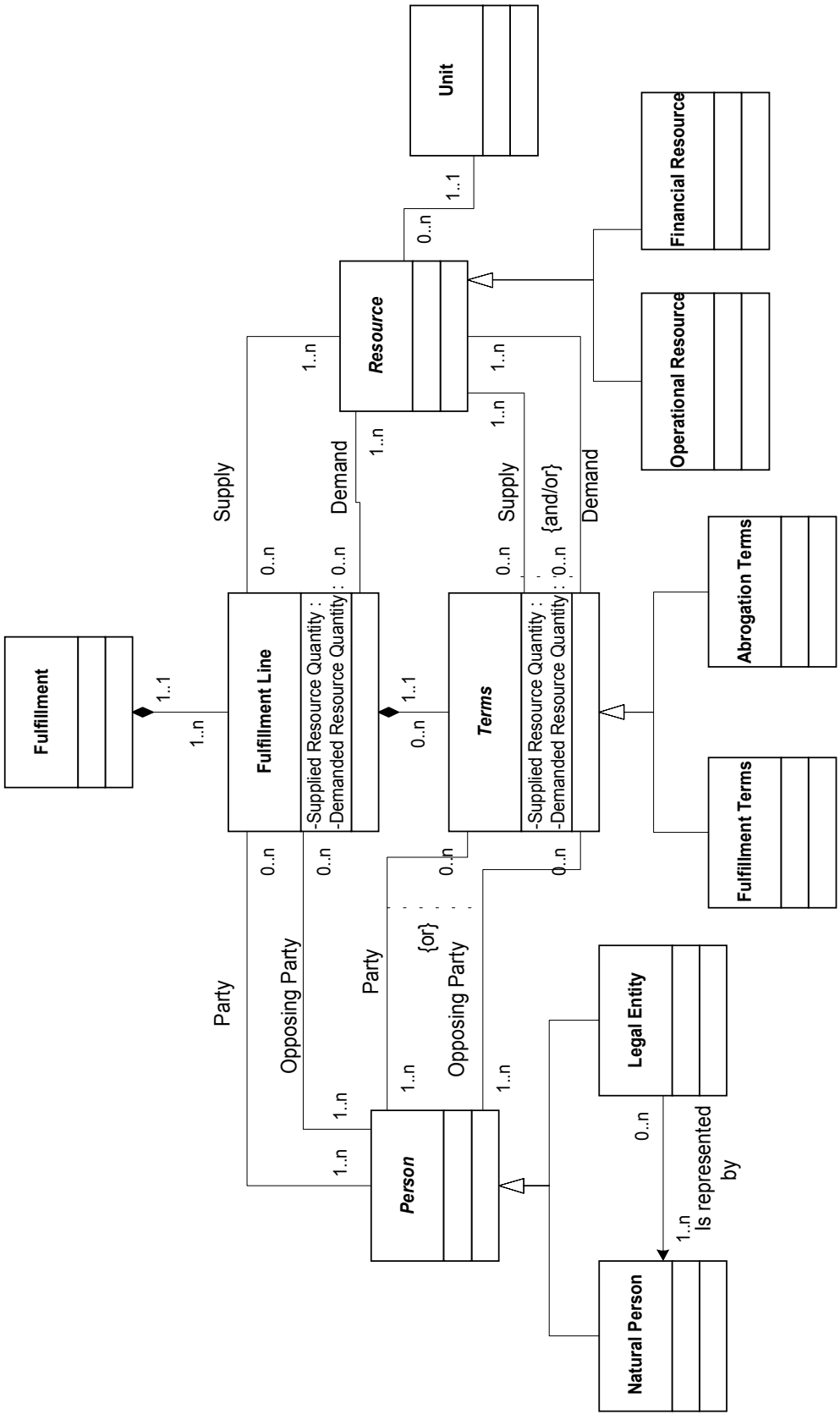


Figure 4-2. Fulfilment model

These relationships were termed as the 'supply' and 'demand' of resources. As explained earlier in this section, fulfilment lines hold information on the execution of contract clauses. There are two types of fulfilment lines. The first type of fulfilment line is used by fulfilments that exclusively focus on the execution of resources used on *one side* of the exchange (the supply side or the demand side). Examples are delivery notes, receipt notes, etc. Supposing a clause details the exchange of products against money. The delivery note is one of the fulfilments used in the execution of this clause but only details the delivery of the resources defined as the subjects of the clause – the products. In the fulfilment line representing the delivery note, no detail is available on the resources offered as compensation (in this case, money). The second type of fulfilment line supports fulfilments detailing the execution of resources on *both sides* of the exchange (the supply side and the demand side). Examples are invoices, cheque payments, etc. Returning to the above example of a clause detailing the exchange of products for money, the invoice details the obligation (or right, depending on the participant's perspective) in terms of the money resource resultant from the delivery of products; in other words, the resource supplied is detailed in terms of the resource demanded. The cheque payment details the supply of money due for products delivered – both resources are detailed. Section 4.2 explained that participants in clauses could be defined as performing either the role of clause 'party' or clause 'opposing party' (Design Feature 2). For fulfilments, the same solution applies, i.e. participants can be cast in the role of fulfilment 'party' or fulfilment 'opposing party'. Design Feature 3 explained which types of participants are available for clauses and how they are termed in legal theory (i.e. 'natural persons' and 'legal entities'). The same solution is implemented for fulfilments and is therefore not repeated here. Design Feature 4 explained that in the Contract Clause Model, conditions under which an exchange takes place are detailed in 'terms'. Two types of terms could be distinguished, the clause terms and the abrogation terms. The conditions under which the fulfilments takes place are also detailed in terms. However, the application of 'terms' for fulfilments is not totally equivalent to the application of terms for clauses. A 'clause' details the definition of a resource exchange and the *entire* execution of this exchange is detailed in one or more terms. For instance, in the exchange of products for money, the terms of delivery and the terms of payment are detailed at a clause level. A fulfilment only contains detail on the execution at one particular step. In other words, in most cases, several fulfilments are required to support the entire execution of a clause. With regard to the application of 'terms' to fulfilments, only the terms applicable to a particular fulfilment are specified on that fulfilment. Either type of term, 'clause terms' or 'abrogation terms', can be defined for a fulfilment. For example, firstly, in a delivery note, the terms of delivery are specified along with the corresponding terms of abrogation; secondly, in an invoice, the terms of payment are specified along with the corresponding terms of abrogation. The same solution as described in Section 4.2 to be applied to contract clauses is implemented in the Fulfilment Model and therefore not repeated here. As distinct from the clause, it is not relevant to define a fulfilment over different stages of its lifecycle. It is also not relevant to maintain a portfolio of 'fulfilments', comparable to the clause portfolio. As a consequence, no relations are maintained between 'fulfilment lines'.

Requirements

The following requirements are supported by the features offered in the Fulfilment Model:

Requirement 1. Data on a single BPI to be defined coherently. This model supports the coherence of data on fulfilments of a BPI. Fulfilments are defined in the [Fulfilment] class and consist of one or more fulfilment lines, defined in the [Fulfilment Line] class. A fulfilment line represents an exchange of resources in the execution phase. This is implemented by the supply and demand associations between the [Fulfilment Line] and the [Resource] classes. In the paragraph describing the functionality of the Fulfilment Model above, it was explained that a fulfilment line can represent both sides of a resource exchange (the supply and demand sides), or only one side of a resource exchange (the supply or the

demand side). This functionality is implemented through further detail on the supply and demand association. This detail consists of the '{and/or}' specification that states that either both or only one of the associations has to be applied. Two further remarks have to be made. First, this model does not support the exchange part of a BPI (this functionality was supported in the Contract Clause Model, see Section 4.2). Second, this model does not support the integration of the fulfilments of a BPI and the exchange (i.e. the clause) part of a BPI. This integration is the subject of the third model (the integration between the 'Contract Clause Model' and the 'Fulfilment Model', resulting in the 'Contract Model', see Section 4.4).

Requirement 2. Data on a BPI is ex post data (the first part of this requirement, that data on a BPI is *ex ante* data, is supported in the Contract Clause Model, see Section 4.2). Fulfilments always pertain to the execution of clauses. They are recorded once a specific step in the execution of a clause has been fulfilled. They consequently always contain *ex post* data. As explained earlier in this paragraph, in the Fulfilment Model, the relationship between a clause and its fulfilments is not yet supported. This functionality is described in Section 4.4 (the Contract Model).

Implementation Choices
The same implementation choices described for the Contract Clause Model are also applicable to the Fulfilment Model. A full explanation is presented in Section 4.2. The Fulfilment Model is illustrated in Figure 4-2.

4.4 Contract Model

Functionality
The Contract Model supports all the requirements of the contract data model as described in the introduction. In this model, the requirements concerning 'contracts' and 'clauses' (as described in Section 4.2) and 'fulfilments' and 'fulfilment lines' (as described in Section 4.3) are integrated. This section only handles functionality additional to the Contract Clause Model and the Fulfilment Model that relates to the integration (matching) between 'contracts' and 'fulfilments'. It is necessary to support matching at sublevels of these documents (i.e. matching between the 'contract clause' and the 'fulfilment line') because a 'fulfilment' (e.g. an invoice) can consist of 'fulfilment lines' (e.g. invoice lines), concerning exchanges (defined in 'clauses') belonging to different 'contracts'. A 'clause' is capable of being fulfilled at different times and is therefore supported by several 'fulfilment lines' (e.g. in an exchange of products for money the products are delivered on several occasions, and therefore several delivery notes are required). Likewise, a single fulfilment line (e.g. an invoice line) can be used to invoice several exchanges (e.g. several purchase order lines are aggregated into one invoice line). The matching between 'fulfilment lines' and 'clauses' is supported by a specific relationship (association) between the [Fulfilment Line] class and the [Clause] class.

Requirements
This paragraph considers which requirements are supported by the Contract Model. In comparison to the paragraph on the functionality of the Contract Model where only additional functionality with regard to the Contract Clause Model and the Fulfilment Model is considered, here the requirements supported by the entire Contract Model are evaluated. The following requirements are supported.

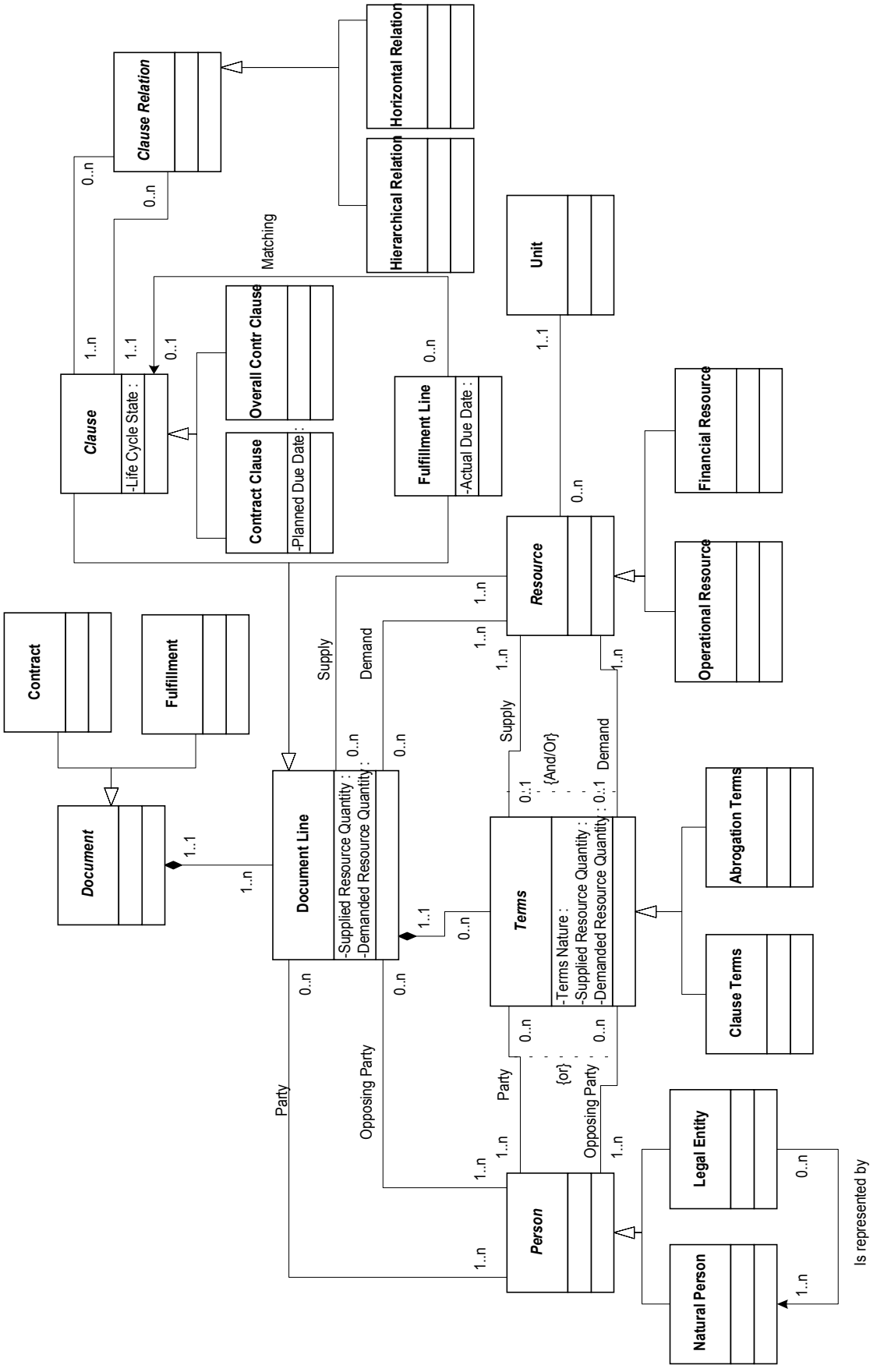


Figure 4-3. Contract model

Requirement 1. Data on a single BPI to be defined coherently. The data required on a BPI consists of two types. The first type concerns data on the *exchange* in different phases of its lifecycle. The exchange is supported by the functionality related to the 'contract clause'. In the Contract Model, exchanges are defined in the [Clause] class, which is a subclass of the [Document Line] class. The supply and demand relationships (associations) are defined between the [Document Line] and [Resource] classes[28]. The second type of data concerns data on the *execution of an exchange*, also referred to as a 'fulfilment'. In the Contract Model, data on the execution of exchanges are defined in the [Fulfilment Line] class (a subclass of the [Document Line] class). As explained in the Fulfilment Model (see Section 4.3), 'fulfilments' can concern resources on either side of an exchange or resources on both sides of an exchange. The supply and demand relationships (associations) are defined between the [Document Line] and [Resource] classes. The coherence between 'clauses' (defined in the [Clause] class) and the 'fulfilment lines' (defined in the [Fulfilment Line] class) is supported through the 'matching' relationship (association) defined between the [Clause] class and the [Fulfilment Line] class.

Requirement 2. Data on different BPIs to be defined coherently. This functionality relates to the necessity to support the requirements of the 'contract portfolio'. This requirement only concerns supporting the possible relationships between 'contract clauses'. The relationship between the overall contract clause and another overall contract clause and the relationship between overall contract clauses and a normal contract clause (i.e. a contract clause defined at the lowest level) are the two defined. There are no possible relationships between 'fulfilment lines'. Two different types of relationships can be defined between clauses, namely hierarchical relationships and horizontal relationships. Hierarchical relationships are defined in the [Hierarchical Relationship] class and horizontal relationships are defined in the [Horizontal Relationship] class.

Requirement 3. Data on BPIs is ex ante *and* ex post *data.* The *ex ante* part of the data concerns the support of resource exchanges over the different lifecycle phases (planned, committed). This functionality is supported in the 'contract clause', along with a 'lifecycle state' property defined in the [Clause] class. The *ex post* part of the data concerns the support of exchange executions. This functionality is supported by 'fulfilment lines', defined in the [Fulfilment Line] class.

4.5 Summary

This chapter adds to the body of knowledge on accounting data model research by proposing an object-oriented domain analysis model for the area of BPI data storage. In Section 2.5 of Chapter 2, the contract principle was identified as an approach suitable for structuring BPI data. Design features with which the contract data model design must comply were later defined in Sections 3.2.6 and 3.3.3 of Chapter 3. A static object model was presented in UML and was built in three phases. The first model is the presentation of the Contract Clause Model, which reflects the capability of capturing data on resource exchanges. The second model concerns fulfilments, which enable data storage on resource exchange execution. The third and final model models the entire Contract Model. This model represents the integration between the Contract Clause Model and the Fulfilment Model. That last model concludes the proposal of the shared data model based on contracts as a workable and improved solution to accommodate data in ERP systems from a shared data environment. The question of whether data can indeed be accommodated using the contract model to service new information

[28] The supply and demand relationships are defined between the [Document Line] and [Resource] classes and not between the [Clause] and [Resource] classes because this behaviour is common to both the Clause and the Fulfilment Line. These association relations are therefore defined in the super class.

requests over time remains unanswered at this point. Part four of this research (see Chapters 5 to 8) will focus on the definition of a new application for hierarchical treasury management decision-making serviced by *ex ante* and *ex post* accounting data. To service this application, it will be evaluated whether or not the Contract Model, as detailed in Section 4.4, can accommodate sufficient data.

Part IV: Validation