CEN/TC XBRL

Date: 2013-06

TC XBRL WI XBRL003

CEN/TC XBRL

Secretariat: NEN

Improving transparency in financial and business reporting — Harmonisation topics — Part 1: European data point methodology for supervisory reporting

Einführendes Element — Haupt-Element — Teil 1: Ergänzendes Element

Élément introductif — Élément central — Partie 1 : Élément complémentaire

ICS:

Descriptors:

Document type: CWA
Document subtype:
Document stage: CEN Enquiry
Document language: E

Contents

Foreword

This document has been prepared by CEN/WS XBRL, the secretariat of which is held by NEN.

CWA XBRL 001 consists of the following parts, under the general title *Improving transparency in financial and business reporting — Harmonisation topics*:

— Part 1: European data point methodology for supervisory reporting

— Part 2: Guidelines for data point modeling

— Part 3: European XBRL Taxonomy Architecture

— Part 4: European Filing Rules

This document is currently submitted to a public consultation.

Introduction

General

Data Point Modelling is a data oriented methodical procedure to create semantic and multidimensional models that reflect the reporting requirements of European supervisors. Reporting requirements are defined by regulations and represented in tables. First Data Point Models (DPM) were developed in 2009 to describe the data in a redundancy-free, consistent and unambiguous way.

Semantic models are used to ease the communication between domain experts and IT specialists. Whereas formal models are defined for technical purposes semantic models are defined from a viewpoint of a domain user. They can contain definitions, documentations and explanations. Domain experts decide which objects are relevant and which relations exists between the objects of the model. Semantic models are independent of any physical implementation.

The characteristic of multidimensional models is the division of data in quantitative and qualitative aspects. Parameters that are measured in figures (also known as metrics) are quantitative aspects that often build the basis of data analyses. Qualitative aspects provide a closer description for these parameters. Data objects based on multidimensional models are referred to as facts. Fact attributes are the quantitative aspects of a fact and dimensions are the synonym of the qualitative aspects of a fact.

A Data Point Model reflects semantic and multidimensional aspects of data modelling. Data Point Models should enhance the understanding of the data requirements for the reporting entities by providing information to correlations that exceed the information given only by table structures. The main challenge in data modelling is the identification of implicit information given in tables and its transformation in a logical model. Data Points Models are typically created by banking specialists who are highly skilled in understanding supervisory reporting frameworks.

This document intends to support the communication between supervisory experts and IT experts by introducing the concept of data point modelling and its underlying terms. Data Point Models remain as semantic models at first technologically-neutral but they are used by IT specialists (1) for generating data formats for the reporting process or (2) for designing multidimensional as well as relational database structures for the analysis of supervisory data.

This guidance is in the form of notes in association with the pertaining requirements clause and uses the terms “MUST” (strong recommendation), “SHOULD” (recommendation) and “MAY” (possibility). Organizations wishing to implement this CWA (CEN Workshop Agreement) would be expected to consider all recommendations where the terms "MUST" and “SHOULD” are used.

Objective

A Data Point Model consists of objects that reflect the supervisory metadata and their relations among each other that can be communicated and understood by computers. The objects of a data point model described in this document facilitate the ease of understanding of the data structure for technicians and reflects the definitions, rules and constraints to be met when using a Data Point Model as basis for the generation of a data format or as basis for analysis purposes.

Target Audience

This document is being created to support Information Technology (IT) experts in the transfer of content from regulatory reporting to IT systems. It assumes that the reader has a working knowledge on multidimensional models. Furthermore basic knowledge about Business Intelligence is assumed to understand the rules to be followed when designing multidimensional database structure for data warehouses.

1 Scope

The Data Point Methodology has been defined for the creation of Data Point Models in the context of European supervisory reporting. Data Point Models are published by an European supervisory authority. To reflect the defined structures in a machine-readable form they can be accompanied by an XBRL taxonomy. It is also possible to extend the described methodology to other environments.

2 Data Point Meta Model

2.1 General

The Data Point Meta Model provides (1) the components for the construction of a formal model that describes sets of DataPoints relevant to European supervisory reporting frameworks, (2) definitions, rules and constraints on how to combine these components and (3) the meaning (semantic) of the components and relations. Similar to a model construction kit for toys it provides the modelling principles with all characteristics available for use by the modeller. A UML class diagram is used to provide the syntax and semantic to define the meta model for data points by showing the relevant classes and their attributes.

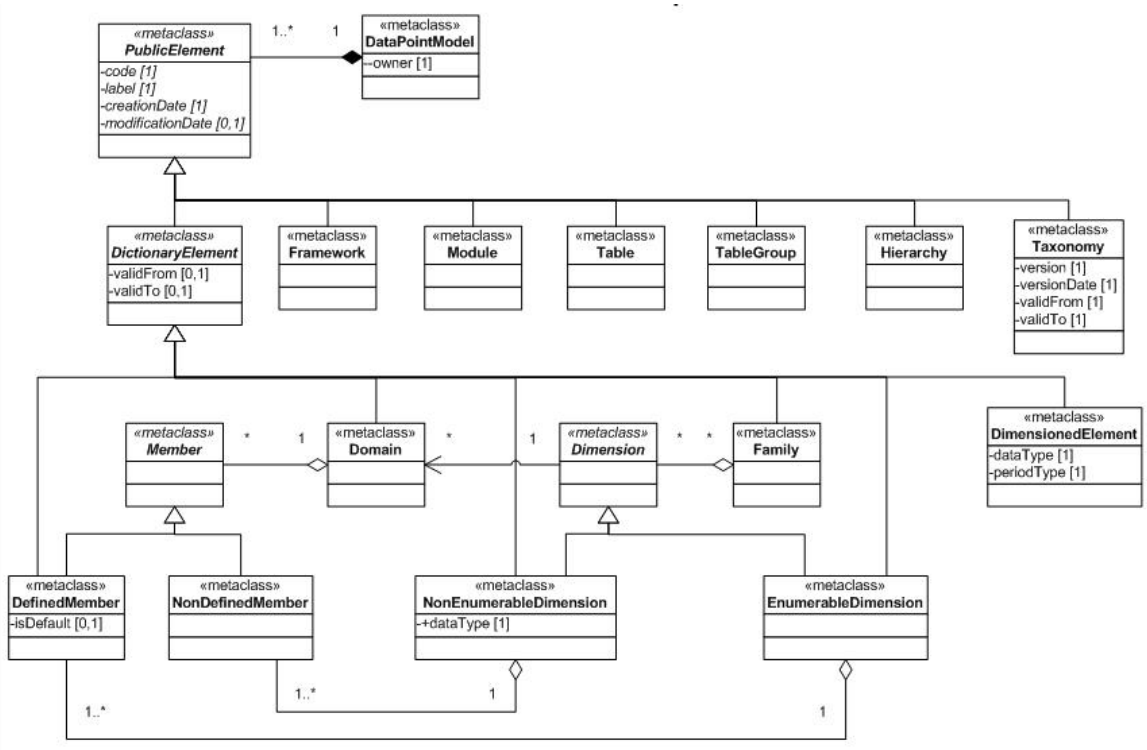

Figure 1 —Structural Perspective

2.2 Definitions

2.2.1 DataPointModel

A DataPointModel defines structures of data describing the characteristics of the information exchanged in the context of supervisory reporting processes. A DataPointModel consists of a dictionary of business concepts and their properties which are represented in tables. It identifies the content of each DataPoint by its granular components with a semantic meaning and its relation to other data points. A DataPointModel has a mandatory attributue owner who needs to be made explicit.

From an IT perspective a DataPointModel can be interpreted by IT applications which enable (1) a generation of data formats for the reporting process or (2) the design of multidimensional database structures for the analysis of supervisory data, i.e., in data warehouses.

Contained elements: *PublicElement*

References:

meta class	multiplicity	description
PublicElement	one or more	References the collection of PublicElements owned by a DataPointModel.

2.2.2 PublicElement

A PublicElement is a generalization of a concept of the model. It is identified by a code and consists of an appropriate label. PublicElements have two additional attributes giving information about the dates of creation and modification. Each PublicElement is owned by an institution or an organization. PublicElements are abstract and need to be specified by their concrete sub classes like Frameworks, Tables etc.

References:

meta class	multiplicity	description
DataPointModel	exactly one	References the DataPointModel owning the collection of PublicElements.

2.2.3 DictionaryElement

DictionaryElement is an abstract class of elements that build the basis of the core concepts of a DataPointModel like DimensionedElements, Dimensions, Domains and DefinedMembers. They are derived from PublicElements and may define a currency period to support versioning and enable a filtering of obsolete elements by applications. The currency period is defined by two optional attributes validFrom and validTo which should ease the maintenance of elements of the DataPointModel in the course of time.

Superclass: *PublicElement*

2.2.4 Framework

A Framework is a business term common to a group of business users that consists of reporting regulations for a domain specific scope of information, like COREP, FINREP or Solvency II.

The information requirements of a Framework are structured through business templates to ease the understanding for the reporting entities that are obliged to submit the information to their supervisor. A business template is a collection of supervisory data ordered in a representation that is fitting to the business domain. On basis of business templates users are able to understand the context and relationships between the required data. Business rules to be met by the reporting entities are also defined in the reporting regulations. Some rules are incorporated in the design of the business templates. E.g. the detailed information which is being part of a summation.

A DataPointModel can refer to one or more supervisory Frameworks. Information should be given to which Framework the defined elements of the DataPointModel refer to.

Superclass: *PublicElement*

2.2.5 Table

A Table reflects the structure of a business template or represent an individual view on supervisory data based on a specific business context. As business templates are used for the communication between supervisors and reporting entities DataPoints are grouped in Tables in a DataPointModel to reconstruct the business templates for presentational purposes.

A DataPointModel thus must contain presentational information to reconstruct theses defined tables.

A Table consists of the combination of one, two or three Axes which form TableColumns (in the X-Axis), TableRows (in the Y-Axis) and TableSheets (in the Z-Axis). A duplication of Tables is indicated by two or more TableSheets. Axes can be built on basis of a set of DictionaryElements that could be already defined in a Hierarchy. The combination of the DictionaryElements in each Axis defines a Cartesian product which represents the set of DataPoints reflected in a Table. Tables are normalized from a dimensional perspective and reflect the design constraint that a DictionaryElement can only be associated to one Axis.

Superclass: *PublicElement*

2.2.6 TableGroup

A TableGroup is a set of Tables that represents a business template. A TableGroup is being created when a business template defines more than one table to reflect the business context. A TableGroup needs also to be created when the business template refers to the same dimension-member combinations in more than one axis. The business template is to be split in two or more Tables to prevent that the same Dimension is associated to a DataPoint more than once.

Superclass: *PublicElement*

2.2.7 Hierarchy

Members as well as DimensionedElements can be arranged in Hierarchies to represent the relationships to one another. In mathematical terms a Hierarchy is a rooted tree that provides the information if a Member is at top level, below another Member or at the same level. Financial information is often split up in different segmental breakdowns which represent Dimensions in multidimensional terms. If the Members of a Dimension share the same level of detail, they could be represented as a flat list. But often the Members relate to each other, i.e. in a parent-child relationship, and so they form natural hierarchies. The information about the location of a Member in a Hierarchy of a Dimension improves its understanding. Furthermore, Hierarchies can be used to define rules for calculations or aggregations. In the DataPointModel a Hierarchy forms are sets of DefinedMembers of an EnumerableDimension arranged in a hierarchical disposition.

Superclass: *PublicElement*

2.2.8 Taxonomy

A Taxonomy combines the components for a version of the DataPointModel for a given period in time. Its currency period is defined by the attributes validFrom and validTo. By creating a relation between Taxonomy and DictionaryElements, Tables and Hierarchies a set of valid DataPointModel components are being created. PublicElements like Table and Hierarchy can be reused when they have not been modified since the last Taxonomy version.

Superclass: *PublicElement*

2.2.9 Module

A Module is a group of DataCubes that carry relevant identical semantics and may serve the reporting process. Modules define sets of business information that should be reported together, i.e. to conduct validation rules that are defined across DataCubes.

Superclass: *PublicElement*

2.2.10 Dimension

A Dimension is a data set to one characteristic area which is composed of individual and non-overlapping data elements. In the context of a DataPointModel Dimensions are used to group information in a meaningful way. Dimensions are used to define "by" conditions and provide structured information to describe a DataPoint in detail. A Dimension can refer to a Domain. Whereas the DimensionedElement represents the quantitative aspects, the qualitative aspects are described by Dimensions. The data elements given to Dimensions are called Members.

Superclass: *DictionaryElement*

References:

meta class	multiplicity	description
Domain	zero or more	References a collection of Domains associated with a Dimension.
Family	zero or more	References the Families owning a collection of Dimensions.

2.2.11 Domain

A Domain is a classification system to categorize items that share a common semantic identity. A Domain provides therefore an unambiguous collection of items in a value range. The items of a Domain can have a definite, and therefore countable, number of items, or an infinite number of elements that follow a specific (syntax) pattern.

Contained elements: *Member*

References:

meta class	multiplicity	description
Dimension	exactly one	References the Dimension associated with this Domain.

2.2.12 EnumerableDimension

An EnumerableDimension is a subclass of Dimension that specifies a finite number of Members.

Superclass: *DictionaryElement, Dimension*

Contained elements: *DefinedMember*

References:

meta class	multiplicity	description
DefinedMember	one or more	References the set of DefinedMembers owned by an EnumerableDimension.

2.2.13 NonEnumerableDimension

A NonEnumerableDimension is a subclass of Dimension that specifies an undefined number of Members in the Dimension. The NonEnumerableDimension defines syntactic constraints on the values of the Members, i.e. a dataType or a specific pattern.

Superclass: *DictionaryElement, Dimension*

Contained elements: *NonDefinedMember*

References:

meta class	multiplicity	description
NonDefinedMember	one or more	References the set of NonDefinedMembers owned by an NonEnumerableDimension.

2.2.14 Member

A Member is the data element that is given to a Dimension. Members can be grouped in Domains. Members in a Domain share certain semantic identity, just like the set of Members in a Dimension.

2.2.14.1 DefinedMember

A DefinedMember is discrete and countable. These Members can be explicitly listed in an enumeration. The metaclass DefinedMember has an optional attribute default.

Superclass: *DictionaryElement, Member*

References:

meta class	multiplicity	description
EnumerableDimension	exactly one	References the EnumerableDimension owning the collection of DefinedMembers.

2.2.14.2 NonDefinedMember

A NonDefinedMember is defined by syntactic constraints on a possible value, not the value itself.

Superclass: *Member*

References:

meta class	multiplicity	description
NonEnumerableDimension	exactly one	References the NonEnumerableDimension owning the collection of NonDefinedMembers.

2.2.15 DimensionedElement

DimensionedElements represent the nature of the data with a fixed and unchangeable meaning. DimensionedElements are strongly related to the underlying data type. Mostly they are numeric and quantitatively measurable to be used for calculations and aggregations but they can be also reflect boolean or date values. A DimensionedElement is the essential part of a DataPoint that can also refer to zero or more Dimensions with its according set of Members.

— The attribute dataType establishes the set of possible values of the facts reported according to that metric: monetary information, percentages, dates, texts…

— The attribute periodType defines whether the property / amount to be measured corresponds to a specific moment in time (instant type) or whether its nature requires it to be obtained by taking measures during an interval of time (duration type).

Superclass: *DictionaryElement*

2.2.16 Family

Families are groups of Dimensions relevant for presentation or querying purposes.

Superclass: *DictionaryElement*

Contained elements: *Dimension*

References:

meta class	multiplicity	description
Dimension	zero or more	References the set of Dimensions owned by a Family.

2.3 Versioning Perspective

2.3.1 Introduction

A DataPointModel combines the reporting regulations for several specific scopes of information (solvency information, financial information…) summarized in one or more Frameworks. The name of a Framework is stable in time but the business templates referring to a Framework change according to amended reporting requirements in the course of time. A Taxonomy represents a set of reporting requirements which are enforced by normative or legal acts for a period of time. The currency period starts with a law becoming effective. In general a new Taxonomy version must replace the previous one so the currency period of the old one ends before the new version becomes valid. It is not recommended to have two overlapping Taxonomy versions referring to the same reporting period.

A Taxonomy consists of DictionaryElements that are valid for the currency period given by the Taxonomy. DictionaryElements which validTo date ended before the currency period start are not part of the Taxonomy. Over the course of time several Taxonomy versions exists which may refer to one Framework. In order to reduce the cost of maintenance, Tables from previously released Taxonomies that have not suffered any modification can be referred to from a new Taxonomy.

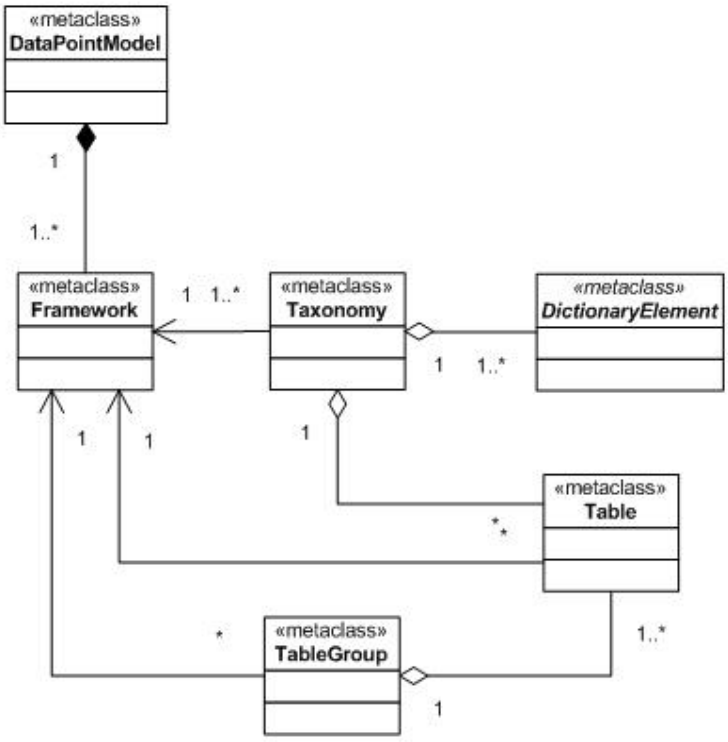

Figure 2 — Versioning perspective

2.3.2 DataPointModel – References

References:

meta class	multiplicity	description
Framework	one or more	References the collection of Frameworks owned by a DataPointModel.

2.3.3 Framework – References

References:

meta class	multiplicity	description
DataPointModel	exactly one	References the DataPointModel owning a collection of Frameworks.
TableGroup	zero or more	References the set of TableGroups associated with this Framework.
Table	zero or more	References the set of Tables associated with this Framework.

2.3.4 Taxonomy – References

References:

meta class	multiplicity	description
DictionaryElement	one or more	References the collection of valid DictionaryElements owned by a Taxonomy.
Table	zero or more	References the collection of Tables owned by a Taxonomy.

2.3.5 DictionaryElement – References

References:

meta class	multiplicity	description
Taxonomy	exactly one	References the Taxonomy owning a collection of valid DictionaryElements.

2.3.6 Table – References

References:

meta class	multiplicity	description
Taxonomy	exactly one	References the Taxonomy owning a collection of Tables.
Framework	exactly one	References the Framework associated with this Table.
TableGroup	exactly one	References the TableGroup owning a collection of Tables.

2.3.7 TableGroup – References

References:

meta class	multiplicity	description
Table	one or more	References the collection of Tables owned by a TableGroup.
Framework	exactly one	References the Framework associated with this TableGroup.

2.4 Dimension Validation Perspective

2.4.1 Introduction

The dimensional validation that takes place on DataPoints is hosted by DataCubes. The dimensional validation is formed by a DimensionedElement combined with at least one Dimension that hosts at least one Member. Each DataPoint MUST be represented in one DataCube. From a dimensional validation perspective there can be no DimensionedElement that has no Dimensions. Eventhough this perspective shows that a DataPoint can only refer to a DimensionedElement without a relation to a Dimension. This is intentional because there is no rule that does not allow non-dimensional DataPoints.

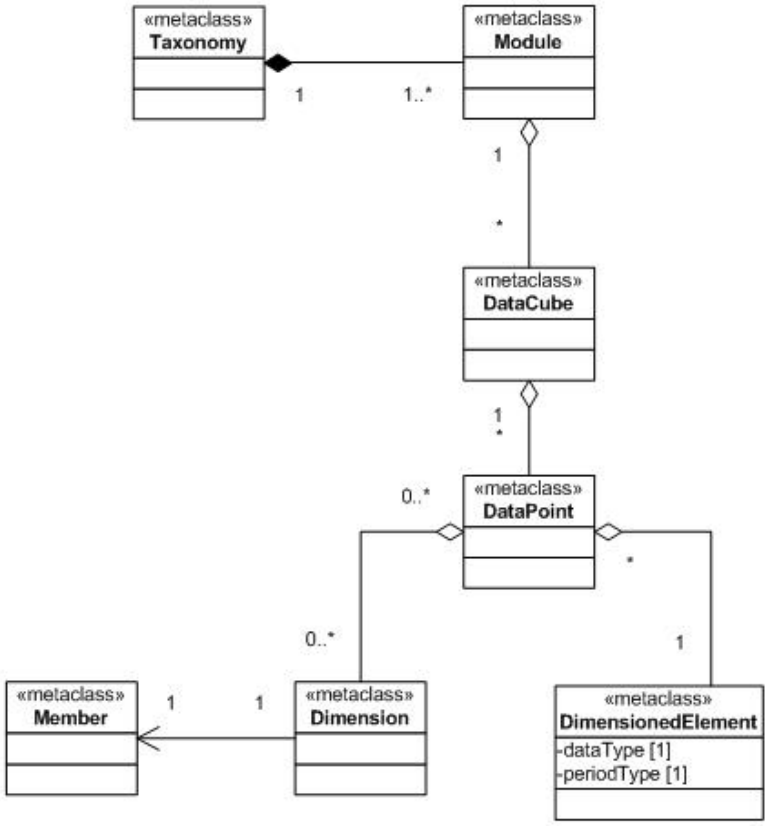

Figure 3 — Dimension Validation Perspective

2.4.2 DataCube

A DataCube is a set of DataPoints with its appropriate Dimensions and Members. A DataCube must be part of at least one Module. A DataCube combines DataPoints that share the same dimensionality. The same dimensionality is given when the DataPoints have the same number of Dimensions and the Dimensions of each DataPoints are equivalent. In comparison to a Table a DataCube must not contain any combination of DictionaryElements that are not allowed. Non allowed data is often marked as gray cells in viewers on business templates.

DataCubes may represent multiple business templates and vice versa, a single business template can be represented in multiple DataCubes, these groups may results in a Module.

Contained elements: *DataPoint*

References:

meta class	multiplicity	description
Module	exactly one	References the Module owning a set of DataCubes.
DataPoint	zero or one	References the collection of DataPoints owned by a DataCube.

2.4.3 DataPoint

A DataPoint is characterized by defining its basic financial meaning and specifying information of breakdowns (Hierarchies) in which it is described in different tables or paragraphs of documentation. A DataPoint can only have one DimensionedElement which holds the quantitative aspects about its dataType (e.g. text, number, percentage) and periodType (i.e. instant, duration). The qualitative aspects of a DataPoint is described by a (set of) Dimension(s) with each Dimension referring to at least one Member.

An EnumerableDimension with a DefinedMember marked as default is implicitly applied when this Dimension is not explicitly associated to a DataPoint. Example: When the DataCube has a dimensional context of 10 Dimensions but only 8 Dimensions are associated with according Members to a DataPoint then two EnumerableDimensions are implicitly included with their default DefinedMembers.

Contained elements: *Dimension*, *DimensionedElement*

References:

meta class	multiplicity	description
DimensionedElement	exactly one	References the element to be dimensioned by a set of Dimensions with their according Members.
Dimension	zero or more	References a set of Dimensions, each Dimension is linked to one Member.

2.4.4 Taxonomy – References

Contained elements: *Module*

References:

meta class	multiplicity	description
Module	one or more	References the collection of Modules owned by a Taxonomy.

2.4.5 Module – References

Contained elements: *DataCube*

References:

meta class	multiplicity	description
DataCube	zero or more	References the collection of DataCubes owned by a Module.

2.4.6 DimensionedElement – References

References:

meta class	multiplicity	description
DataPoint	zero or more	References the set of DataPoints owning the DimensionedElement.

2.4.7 Dimension – References

References:

meta class	multiplicity	description
DataPoint	zero or more	References the DataPoint owning a collection of Dimensions.
Member	exactly one	References the Memeber associated with this Dimension.

2.4.8 Member – References

References:

meta class	multiplicity	description
Dimension	zero or more	References the Dimension associated with this Member.

2.5 Hierarchical Perspective

Hierarchies are brought to the DataPointModel to serve different purposes and have three objectives:

1) Hierachies facilitate validation on DataPoints by defining logical and/or arithmetical relations between DictionaryElements (Eg. a total comprises of details).

2) Hierarchies provide information about how the DictionaryElements should be structured and visualised in a Table view.

3) Hierarchies increase the comprehensibility of DataPoints and their relation to each other. They ease also the maintenance of the DataPointModel.

Hierarchies are optional for DataPointModels. Nevertheless modelling experts advise to add each DefinedMember to a Hierarchy

For arithmetical relationships two warnings are in place:

1) When using multiple DimensionedElements on a single Dimension that has a Hierarchy in its DefinedMembers, the required math may not be possible to perform.

2) When using non-numeric typed DimensionedElements on a single Dimension that has a Hierarchy in its DefinedMembers, the required math may not be possible to perform.

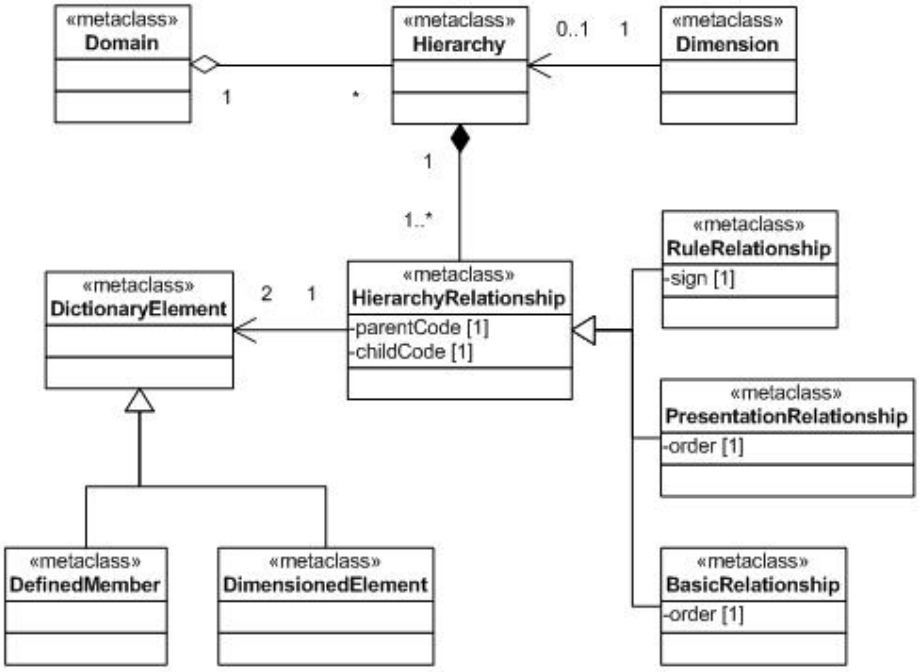

Figure 4 — Hierarchical perspective

2.5.1 HierarchyRelationship

A HierarchyRelationship defines a logical relationship between a pair of DictionaryElements. One of these DictionaryElements defines the parent and the second defines the child. Both DictionaryElements are referenced by their identifiers. A HierarchyRelationship can be of the type presentation, basic or rule and combines DictionaryElements of the same sub class.

References:

meta class	multiplicity	description
Hierarchy	exactly one	References the hierarchy owning a collection of HierarchyRelationships.
Dictionary Element	two	References a pair of Dictionary elements of the same sub class.

2.5.2 RuleRelationship

A RuleRelationship is expressing a mathematical relationship between DimensionedElements. RuleRelationships have a sign attribute which identifies the arithmetical relationship between the two nested elements. The list of possible signs in a DataPointModel is not determined. Examples are: + (plus sign) or - (minus sign).

Superclass: *HierarchyRelationship*

2.5.3 PresentationRelationship

A PresentationRelationship is a Hierarchy for presentational purposes. The order attribute provides information about the ordering of childcodes within the same parentcode. All PresentationRelationships together form a tree structure to be visualized in (e.g.) an Axis of a Table.

Superclass: *HierarchyRelationship*

2.5.4 BasicRelationship

A BasicRelationship is a Hierarchy for indicating semantic relationships that help forming the definition and therefore the comprehension of the structure that is present between DictionaryElements. This eases the maintenance of the DataPointModel.

Superclass: *HierarchyRelationship*

2.5.5 Hierarchy – References

Contained elements: *HierarchyRelationship*

References:

meta class	multiplicity	description
Domain	exactly one	References the Domain owning the collection of Hierarchies.
Family	exactly one	References the Family owning a collection of Hierarchies.
Dimension	exactly one	References the Dimension associated with a Hierarchy.

2.5.6 Dimension – References

References:

meta class	multiplicity	description
Hierarchy	zero or one	References the Hierarchy associated to this Dimension.

2.5.7 Domain – References

Contained elements: *Hierarchy*

References:

meta class	multiplicity	description
Hierarchy	zero or more	References the collection of Hierarchies consisting of DefinedMembers owned by a Domain.

2.5.8 Family – References

Contained elements: *Hierarchy*

References:

meta class	multiplicity	description
Hierarchy	zero or more	References the collection of Hierarchies consisting of Dimensions owned by a Family.

2.6 Presentation Perspective

2.6.1 Introduction

Data required by supervisors is described in legal normative acts and mostly reflected in bi-dimensional forms usually referred to as business templates. In most of the cases a business template is represented by only one Table. Sometimes such a convenient match is not possible because there is a high degree of complexity in the Template that does not allow grouping the DataPoints in the same view as the predefined Template. A split of a business template in different Tables is needed from a presentation perspective. A set of Tables reflects then the data requirements defined in one business template. The DataPointModel reflects the link between the business templates and the Tables that represent their content.

A Table consists of a combination of one or more axes. A single axis 'table' is represented as a flat list of concepts, no facts present. When facts need to be entered or presented a second axis is necessary. To each axis a set of DictionaryElements is assigned to. The X-axis defines the TableColumns as horizontal arrangement whereas the Y-axis represents a vertical progression of TableRows in a Table. The Z-axis can be interpreted as the identifier of a TableSheet in a series of two-dimensional Tables with the same structure. An Axis can be also linked to one or more Hierarchies build upon DictionaryElements.

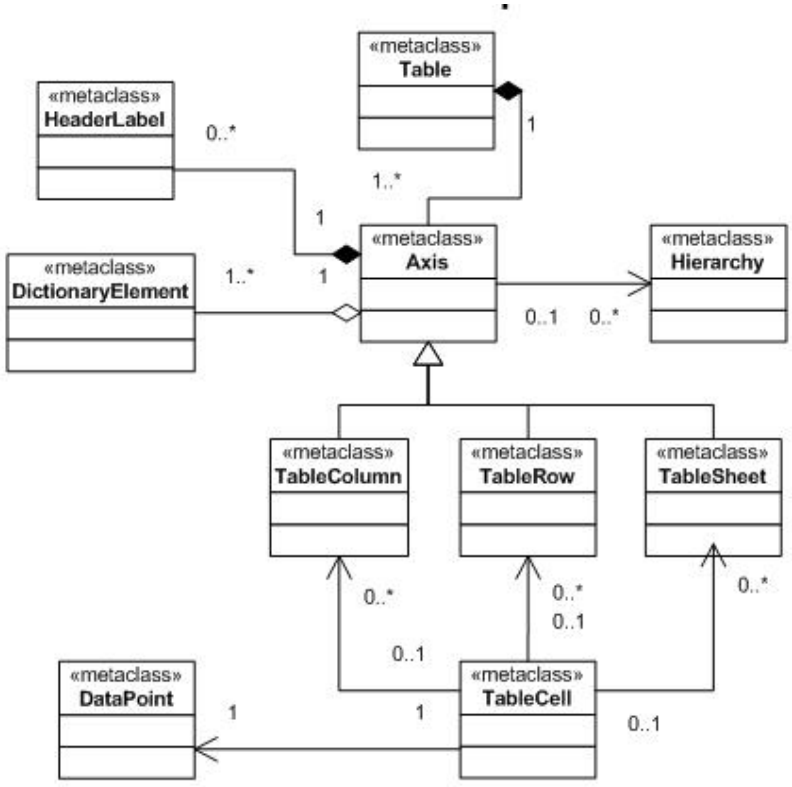

Figure 5 — Presentation perspective

It is possible to assign HeaderLabels to an Axis. They are specific to a Table and only used for presentation purposes. One TableCell in a Table is a combination of one TableColumn, one TableRow and optional TableSheets and inherits the dimensional combinations of the DictionaryElements linked with these axes. A TableCell can represent a DataPoint in the model if it corresponds to data requirements. In the case of the TableCell is not asked by a supervisor a dimensional validation ensures that the incorrect DataPoint is being identified.

2.6.2 Axis

An Axis is considered the vertical or horizontal line that makes up the quadrants of a coordinate plane. The vertical Axis is usually referred to as the Y-axis and the horizontal Axis is usually referred to as the X-axis. In dimensional modelling the Z-axis is considered to represent anything that doesn't apply to the X- or Y-axis. In DataPointModel the Z-axis can best understood as the tabular sheets in a spreadsheet program, representing multiple slices of what the X- and Y-axis display. Often Z-axis content is presented as fixed parameters to the display of X and Y, usually represented in the graph header(s) and footer(s).

Contained elements: *HeaderLabel, DictionaryElement*

References:

meta class	multiplicity	description
Hierarchy	one or more	References the Hierarchy associated with this Axis.
DictionaryElement	one or more	References the collection of DictionaryElements owned by an Axis.
HeaderLabel	zero or more	References the collection of HeaderLabels owned by an Axis.

2.6.3 TableRow

A TableRow is the representation of what the Y-axis is made up of: Each of the individual ordinates is represented on a single TableRow.

Superclass: *Axis*

References:

meta class	multiplicity	description
TableCell	zero or one	References the TableCell associated with this TableRow.

2.6.4 TableColumn

A TableColumn is the representation of what the X-axis is made up of: Each of the individual ordinates is represented on a single TableColumn.

Superclass: *Axis*

References:

meta class	multiplicity	description
TableCell	zero or one	References the TableCell associated with this TableColumn.

2.6.5 TableSheet

A TableSheet is the representation of what the Z-axis is made up of: Each of the individual ordinates CAN be represented on a single TableSheet.

Superclass: *Axis*

References:

meta class	multiplicity	description
TableCell	zero or one	References the TableCell associated with this TableSheet.

2.6.6 TableCell

A TableCell is the coordinate where the ordinates of X and Y axes meet. It represents a place where a single (fact) value can be shown or entered.

References:

meta class	multiplicity	description
TableRow	zero or more	References the TableRow associated with this TableCell.
TableColumn	zero or more	References the TableColumn associated with this TableCell.
TableSheet	zero or more	References the TableSheet associated with this TableCell.
DataPoint	exactly one	References the DataPoint associated with this TableCell.

2.6.7 HeaderLabel

A HeaderLabel is an additional label that is defined on an Axis to allow a grouping of different ordinates under a common headline.

Contained elements: *Axis*

References:

meta class	multiplicity	description
Axis	exactly one	References the Axis owning a collection of HeaderLabels

2.6.8 Table – References

Contained elements: *Axis*

References:

meta class	multiplicity	description
Axis	two or more	References the collection of Axis owned by a Table.

2.6.9 DictionaryElement – References

References:

meta class	multiplicity	description

Axis	exactly one	References the Axis owning a collection of DictionaryElements.

2.6.10 Hierarchy – References

References:

meta class	multiplicity	description
Axis	zero or one	References the Axis associated with this Hierarchy.

2.6.11 DataPoint – References

References:

meta class	multiplicity	description
TableCell	exactly one	References the TableCell associated with this DataPoint.

2.7 Data Point Metamodel Constraints

2.7.1 General constraints

Rule 1.1 — Each PublicElement MUST have a code.
For each PublicElement a technical code MUST be defined.

```
context PublicElement inv:
   self.code->size() = 1
```

Rule 1.2 — All PublicElements belonging to a DataPointModel MUST have unique codes.
Using a code on more than one PublicElement is not allowed.

```
context PublicElement inv:
   self.allInstances()->isUnique(p : PublicElement | p.code)
```

Rule 1.3 — Each PublicElement MUST have at least one label.
At least one Label for a PublicElement MUST be given which provides the human readable meaning of this element.

```
context PublicElement inv:
   self.label->size() >= 1
```

Rule 1.4 — All labels of the set of PublicElements MUST be unique.
Creating more than one label refering to two different PublicElements is not allowed.

```
context PublicElement inv:
   self.allInstances()->isUnique(p : PublicElement | p.label)
```

Rule 1.5 — Each DimensionedElement MUST define a data type and a period type.
Each DimensionedElement is to be determined by a data type and a period type.

```
context DimensionedElement inv:
   self.dataType->size() = 1
   and self.periodType->size() = 1
```

Rule 1.6 — Each DefinedMember MUST be referenced by an EnumerableDimension.
DefinedMembers need a reference to an EnumerableDimension so that they are able to be used for the definition of DataPoints.

```
context DataPointModel inv:
   self.DefinedMember->forAll(m | m.EnumerableDimension->notEmpty())
```

Rule 1.7 — Each DefinedMember SHOULD be part of a Hierarchy.
In some cases breakdowns represent certain business notion rather than disaggregation (e.g. Solo, IFRS consolidation, CRD consolidation). In such case the Hierarchy would be just a flat list of Members.

Rule 1.8 — A DefinedMember MUST be unique in a Hierarchy.
The same Member MUST NOT be used twice in the same Hierarchy.

Rule 1.9 — There MUST NOT be a doubling of DefinedMembers in the same Dimension.
A DefinedMember MUST only be references once in a Dimension.

Rule 1.10 — Each Hierarchy MUST be build on basis of set of HierarchyRelationships of the same sub class.
Hierarchies that serves a specific purpose should be defined by a set of common HierarchyRelationships. I.e. an aggregation should be based on a set of RuleRelationships.

Rule 1.11 — Each Hierarchy SHOULD give information about the Dimension that refer to it.
It eases the identification of the correct hierarchy in a maintenance and impelementation process when a Hierarchy holds information about the Dimension that links to this hierarchy.

Rule 1.12 — Each DataPoint MUST be unique.
There MUST NOT be two data points with the same DimensionedElement, period and dimensional combinations.

Rule 1.13 — Valid DimensionedElements, DefinedMembers and Dimensions MUST be referenced by at least one DataPoint.
Valid DimensionedElements, DefinedMembers and Dimensions MUST be referenced by DataPoints defined in a Taxonomy with a matching currency period.

2.7.2 Data warehouse specific constraints

Rule 1.14 — Each EnumerableDimension MUST refer to at least one Hierarchy.

The DefinedMembers of an EnumerableDimension should be structured by parent-child relationships. In case a natural hierarchy cannot be derived a flat hierarchy should be created.

Rule 1.15 — Each Hierarchy MUST have only one root Member.
In case a natural root in the collection of DefinedMembers can be derived, the DefinedMember set as default should be defined as the root.

Rule 1.16 — Hierarchies that represent aggregations by using RuleRelationships MUST NOT contain minus signs.
In a data warehouse hierarchies are used to enable aggregations across members of dimensions. Minus signs in hierarchies prevent the usage of such a hierarchy in a data warehouse.

2.7.3 European Taxonomy Architecture (ETA) specific constraints

Rule 1.17 — Each Member MUST be referenced by a Domain.

Members need to be listed in Domains. Only Domains can be referenced by Dimensions.

```
context DataPointModel inv:
   self.Member->forAll(m | m.Domain->notEmpty())
```

Rule 1.18 — Each Domain SHOULD have one Member where the attribute *default* is set.
For each EnumerableDimension that refers to this Domain the defined default for this Domain is applied. Together with rule 1.20 this rule ensures that each EnumerableDimension refers to a default which will be applied when the EnumerableDimension is not explicitly set for a DataPoint.

```
context Domain inv:
   self.DefinedMember->select(isDefault = true)->size() = 1
```

Rule 1.19 — Each Dimension MUST point to one Domain.
A Dimension holds the information about the referenced Domain. A Dimension can only be linked with one Domain.

Rule 1.20 — There MUST NOT be a doubling of Members in different Domains.
All DefinedMembers should be consistently defined. A DefinedMember holding a specific semantic meaning must not occur in different Domains.