DRAFT

PROPOSAL OF AN XBRL "PROFILE"

FOR COMMON EUROPEAN REPORTING ON SOLVENCY RATIOS[1]

(January 2005)

This note aims to describe the Banca d'Italia proposal for the XBRL "profile" to be used in the "translation" in XBRL terms of the common European reporting on solvency ratios and, in perspective, of the other reporting schemas produced in the CEBS context. The criteria to whom the "profile" should adhere are clarified (paragraph 1); the description of the data modelling rules to whom the "profile" should adhere follows (paragraph 2); the description of the proposed "profile" is given (paragraph 3); finally some conclusions are drawn, focused on suggestions about possible actions by the "translation" project group and, in the next future, by European supervisors that would decide to adopt the XBRL "product" developed by the project, possibly customising it (paragraph 4).

Of course the note has to be intended as a technical document, whose aim is to facilitate the work of the "translation" project group and whose content can be changed within the project itself.

Document conventions

This section introduces the "typography" used in the document.

Terms representing UML class names, property names and relationship type names are displayed in bold. Class names start with an upper case character; property names and relationship type names start with a lower case character: for example, **Domain** is the name of the UML Domain class; **isMeasure** is a property name; **composedBy** is a relationship type name.

Terms representing a generic instance of an UML class are represented with the same UML class name displayed in normal text: for example Domain is a generic instance of the **Domain** class. Plural nouns are used to describe collections of generic instances of UML classes: for example, DomainElements is the collection of all instances of the **DomainElement** class.

Specific instances of UML classes are displayed in italic underlined text: for example, *geographic location* is a specific instance of the class **Domain**.

Examples are highlighted with a box surrounding the text.

Terms that refer to XML Schema constructs are prefixed by the XSD acronym: for example, XSD complexType is the complexType construct as defined into the XML Schema specification.

1. THE "PROFILE" IDENTIFICATION CRITERIA

According to Banca d'Italia vision, the "profile" MUST or SHOULD meet the following criteria:

- it MUST allow, for the time being, the definition in XBRL terms of the common European reporting on solvency ratios, and, in perspective, of all the other reporting schemas defined in the CEBS context.

  This way, the "profile" should play the role of a unique XBRL format able to transport all data of interest for European supervisors. That would avoid "stovepipe" solutions and would optimise the software development for both the central institutions and the reporting firms, thus being a driver for streamline, i.e. reduce, reporting costs for firms;

[1] Written by P. Milani, F. Di Giovanni, P. Maurizi, M. Romanelli.

- it MUST allow each national supervisor to freely customise, according to COREP rules, an XBRL taxonomy conformant to the "profile" and decide the level of detail desired on every "dimension" present in the templates that constitute the user view of a reporting schema.

  This is a COREP aim;

- it MUST lead in any case to "manageable" taxonomies, i.e. taxonomies that are efficient from the data administration perspective.

  This is considered a sensible and someway "implicit" aim, that should be shared by any supervisor;

- it MUST be fully compatible with the XBRL v2.1 specification (SPEC).

  Otherwise taxonomies based on the "profile" could not be renown by XBRL International as "true" XBRL taxonomies;

- it SHOULD be compatible with the FRTA v1.0 specification (FRTA).

  Otherwise taxonomies based on the "profile" could not be renown by XBRL International as "true" XBRL taxonomies. However, this criterion is deemed to be less "strong" than the previous one (i.e. SHOULD instead of MUST);

- it SHOULD be effective in the description of a reporting schema.

  The more the "profile" strength in the description of a reporting schema, the less the need to supplement the formal (i.e. XBRL) description with other information, probably based on paper;

- it SHOULD favour the optimisation of software development (e.g. maximising the use of off-the-shelf XML Schema validation tools).

  This criterion goes into the direction of a concrete reduction of primary reporting costs;

- it SHOULD be compatible with an SDMX solution.

  This is an aim of the "translation" project.

2. THE DATA MODELLING RULES

According to Banca d'Italia vision, it is deemed important to formally define the modelling rules to whom the chosen XBRL "profile" should adhere.

These modelling rules are briefly listed in plain text and formally described, although at a high level, using the UML technique. They use a language taken from mathematics and algebra, and describe data as mathematical functions. This choice is deemed to confer "robustness" to the model; moreover, it is worth noting that also the relational model is strongly based on assertions taken from mathematics and that this kind of terminology is widely used by those international institutions or organisations mostly active in metamodelling. In short, the proposed rules:

- model data as multi-dimensional "cubes", using the abstract concepts of "variable", "domain" and "domain element";
- someway model the integrity constraints[2] that apply on data.

In the description that follows UML classes and properties relevant for the present work are described. Other aspects of the overall "ideal" model, specifically those that do not fit with the

[2] Integrity constraints are represented by formal, logical or mathematical relationships that can be identified within and between the data.

"translation" project objectives or that are deemed not expressible in XBRL terms, are omitted[3] or only briefly described[4].

2.1 Domain

The class **Domain** is a category of the real world relevant in the given business context.

For example *geographic location*, *sector of economic activity*, *currency* and so on; within COREP context an example of domain is "Exposure class".

The main properties of a Domain are:

- **name** – is the Domain unique identifier;
- **description** – is a human-readable text label.

Other aspects of the Domain description, useful to add semantics (e.g. its algebraic structure, i.e. the definition of allowed mathematical and/or logical operations) have not been represented.

2.2 DomainElement

The class **DomainElement** is an element of a category.

For example *Italy*, *Spain* and so on are elements of the *geographic location* Domain.

Each DomainElement has a **value** and may have a **description**. Other possible properties have not been represented.

Many relationship types can be defined among DomainElements. The **parent-child**, that describes hierarchical relationships between DomainElements, is highlighted. Other types could be handled, like e.g. "synonyms" type or all types derived from the set theory, but they have not been modelled.

2.3 DomainSubset

The class **DomainSubset** is a set of DomainElements belonging to the same category (i.e. Domain). The DomainSubset can be composed of a finite or infinite number of DomainElements. The DomainSubset can be coded or uncoded: elements of coded DomainSubsets are explicitly enumerated; elements of uncoded DomainSubsets are determined by means of a rule.

An example of DomainSubset is *European geographic location* that contains all the States of Europe - i.e. France, England, and so on but not USA, etc. Within COREP, an example of DomainSubset is "Exposure classes for SA Capital requirements & CRM".

The main properties of a DomainSubset are:

- **name** – is the DomainSubset unique identifier;
- **description** – is a human-readable text label;
- **isCoded** – boolean property; its value is "true" if DomainSubset is coded;
- **isFiniteSet** – boolean property; its value is "true" if there exists only a finite number of elements belonging to the DomainSubset;

[3] For example, as the context of interest refers to reporting, other modelling rules could be added that specifically refer to the reporting process, from the perspectives of process description and monitoring. However this part of the model has not been produced, because of the "translation" project objectives.

[4] For example, integrity constraints have been modelled for the formal part only, because the SPEC seems not to be able to fully describe in any case logical and mathematical relationships within data.

- **isFullSet** – boolean property; its value is "true" only for a DomainSubset that contains all the DomainElements. The term FullSet is used in this document for DomainSubset having isFullSet=true.

The **IsSubsetOf** self-relationship type relates a FullSet with all its proper subsets (i.e. a DomainSubset having isFullset=false). The relationship type **refersTo** describes the link between the DomainSubset and the referred Domain. The relationship type **belongsTo** describes the link between the DomainElement and the DomainSubset it belongs to.

2.4 Variable

The class **Variable** is an observable characteristic of the real world relevant in the given business context. This observable characteristic can assume any value drawn by a Domain.

Examples of Variables are *state of birth*, *state of residence*, *counterpart sector* and so on.

State of birth and *state of residence* can assume values drawn by *geographic location* Domain (i.e. Italy, Spain, etc.); *counterpart sector* can assume values drawn by *sector of economic activity* Domain.

Within COREP an example of Variable could be "Exposure class". It can assume values drawn from the "Exposure class" Domain. Note that a Variable and a Domain can have the same name without generating confusion.

Formally speaking, a Variable is defined on a Domain (relationship type **definedOn**) and can host, as its values, any DomainElement belonging to that Domain.

The properties of a Variable are:

- **name** – is the Variable unique identifier;
- **description** – is a human-readable text label;
- **Domain.name** – is the link to the Domain to whom the Variable refers.

Many relationship types can be defined among Variables, like e.g. the "synonyms" type, that describes synonyms between Variables, or the "functional dependency" type, where the value of a Variable depends on the value of another one. They have not been modelled.

2.5 DataCube

The class **DataCube** is a multi-dimensional data, i.e. a data with a classification structure.

Formally speaking, a dataCube is a mathematical function that is defined in an n-dimensional hyperspace (domain of the function) and that takes values from an m-dimensional hyperspace (co-domain of the function). More specifically, the classification structure of a DataCube is represented by the dimensions of the hyperspace where the DataCube (i.e. the function) is defined.

The properties of a DataCube are:

- **name –** is the DataCube unique identifier;
- **description** – is the human readable text label;
- **type** - useful to categorise the DataCube (e.g. time series, or whatever else).

2.6 DataElement

The class **DataElement** is used to formally describe the multi-dimensional structure of a DataCube, i.e. the collection of independent Variables, usually called dimensions and that represent its

classification structure, the collection of dependent Variables, usually called measures, and the collection of the Variables that qualify the measure, usually called attributes.

The **DataElement** class is a couple Variable/DomainSubset, as defined by the the relationship type **hasVariable** with class **Variable** and the relationship type **hasDomainSubset** with class **DomainSubset.**

A DataElement has at least the following properties (the list is potentially open ended):

- **name** - is the DataElement unique identifier;
- **variable** – is a reference to the name of the Variable participating to the couple;
- **useDomain** – is the reference to the name of the DomainSubset that restricts the set of allowable values that the Variable can assume;
- **isMeasure; isAttribute; isDimension** – they identify the role played by the Variable in the DataCube;
- **isTimeDimension** – identifies the Time-Dimension[5];
- **isEntityDimension** – identifies the dimension that represents the entity to whom the data refers[5];
- **isUnit; isDecimal; isPrecision** – they identify specific attributes of the measure necessary to correctly understand the data[5];
- other is-a elements can be added, to better qualify a DataElement.

The **qualify** self-relationship type is showed in order to highlight links between each measure and its proper attributes (e.g. when a DataCube has more than one measure then it is necessary to attach the proper unit, decimal or precision to each measure). Other relationship types could be defined, like e.g. the "functional dependency" type, for which the value of a Variable in a DataElement depends from the value of another one. They have not been modelled.

Within COREP an example of DataCube could be: " SA - Fully Adjusted Exposure value" into the template "STA - Details of exposure value & CRM". This DataCube has four dimensions ("Time", "Exposure class", "Exposure type" and "Reporting institution"), one measure ("Fully adjusted exposure value") and, of course, a series of qualification attributes like unit, decimals and precision.

2.7 CompatibilityRule

The class **CompatibilityRule** provides a way to specify restrictions on the set of admissible combinations of values assumed by Variables within a DataCube[6]. Formally speaking, a CompatibilityRule for a DataCube is a constraint among the Variables belonging to the classification structure of that DataCube.

The **CompatibilityRule** properties are:

- **name** - is the CompatibilityRule unique identifier;
- **description** - is a human-readable text label;
- **Variable.name[n]** - are the links to the set of Variables involved in the CompatibilityRule;
- **value[n]** - define the combination of values of the referenced Variables, where value[i] is the value for the Variable linked by the property Variable.name[i];

[5] These specific properties are highlighted in the model due to the relevance they have in the SPEC.
[6] The CompatibilityRule is a powerful mechanism to be used during the validation process, to verify allowable cross-dimension values.

- **includeExclude** - boolean. It is "true" when the given combination is allowed; it is "false" when the given combination is disallowed .

Within COREP, many examples of CompatibilityRule exist into different templates. These CompatibilityRules, representing forbidden combination of values, are highlighted as black cells and are further explained in narrative footnotes.

The relationship type **involve** provides the link between CompatibilyRule and the given Variables.

2.8 Integrity constraints

Many integrity constraints can be defined that would be used to check the data. Some checks validate the internal structure of a DataCube, some others validate the coherence among different DataCubes.

Examples of the first type of integrity constraints are:

- "structural" checks, whose aim is to verify that each DataCube has all and only the expected dimensions, measures and attributes;
- "formal" checks on each DataElement, whose aim is to verify that the assumed value belongs to the DomainSubset on which the DataElement itself is defined;
- "cross-dimension" checks, whose aim is to verify that the combination of DataElements values is admissible.

Examples of the second type of integrity constraints are:

- "comparison for equality". For instance, in case of a DataCube that is an aggregation of other DataCubes, that means to verify that the measure of the first DataCube is equal to the sum of the measures of the other DataCubes;
- "comparison for nearly-equality" (i.e. the same as the previous bullet, except that the test for equality is accomplished considering equal values that differ for less than a fixed threshold).

Specific relationship types and properties are highlighted in the UML diagram to deal with the aforementioned issues. In short:

- constraints on the overall structure are described by the **hasStructure** relationship type, linking a DataCube to its Structure;
- constraints on values of each DataElement are described by the **useDomain** property of the **DataElement** class;
- constraints on combinations of values are described by the **hasCompatibiliyRule** relationship type, linking a DataCube to the proper CompatibilityRule;
- constraints involving two or more DataCubes are at the moment described by the **composedBy** self-relationship type only. In fact, the complete description of this portion of the model, that would also imply the representation of calculation rules on each dimension of the DataCube structure, goes beyond the present work, because Banca d'Italia deems it not expressible in XBRL terms using the SPEC.

2.9 Presentation

The **Presentation** class is a set of rules useful to rearrange the sequence of DataCubes, in order to build a user view of the information to be reported. In detail, the Presentation contains rules to define hierarchical organisation of DataCubes and ordering between siblings.

The **hasUserView** relationship type links a DataCube to those Presentation rules.

2.10 Document

The class **Document** represents every external resource (files, web sites, normative documentation, extended descriptions, etc.) useful to describe or add semantics to the model objects.

A Document has at least the following properties (the list is potentially open ended):

- **name** - is the Document unique identifier;
- **description** - is a human-readable text label;
- **isNote** – boolean property; its value is "true" if the Document is a textual note;
- **isLink** - boolean property; its value is "true" if the Document is an URL;
- **isDocument** - boolean property; its value is "true" if the Document is a file;
- other is-a properties can be added, to further qualify a Document.

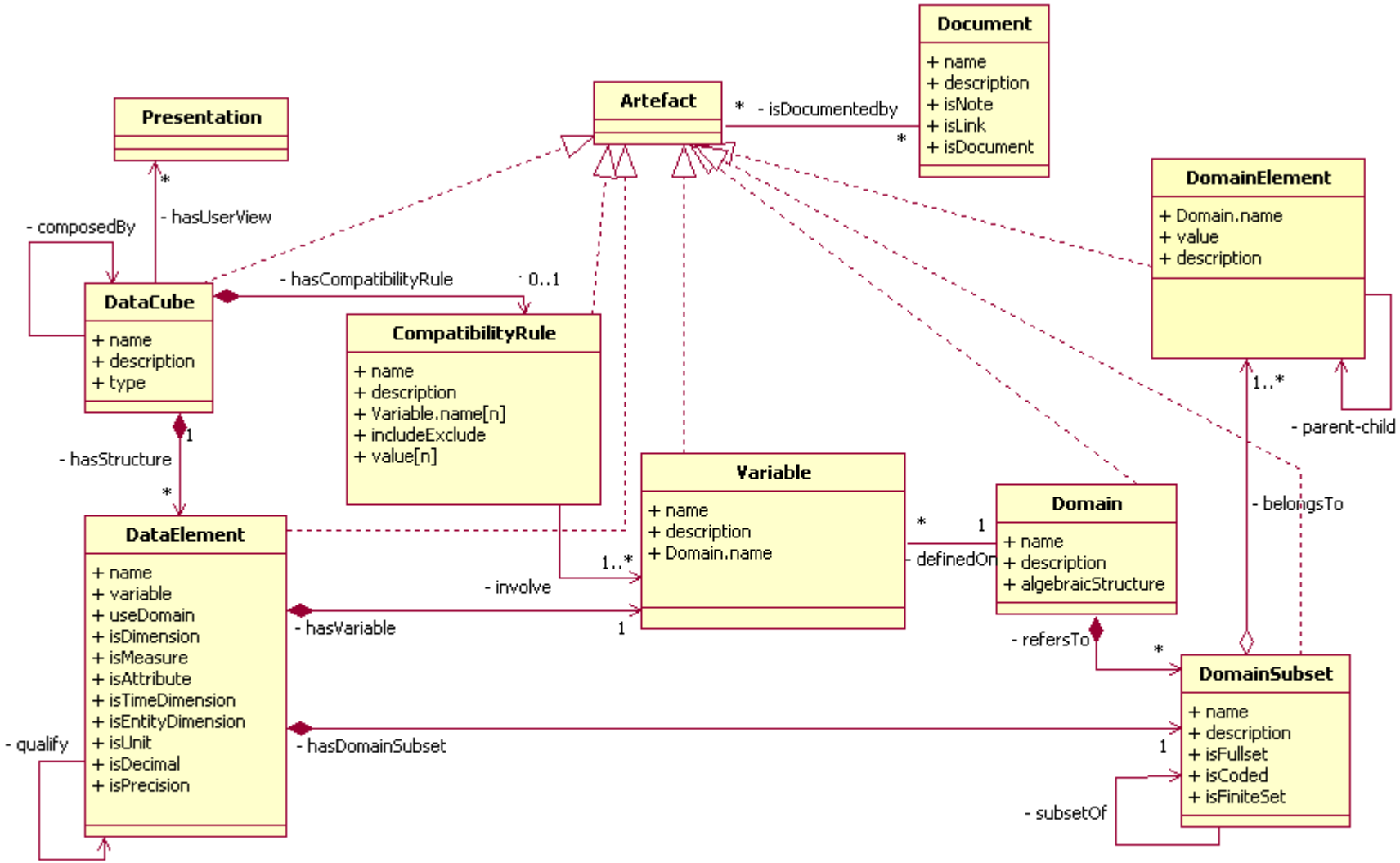
2.11 Artefact

The class **Artefact** is an abstract one and it is ancestor of all other classes showed in the UML diagram. Its only purpose is to describe, in a unique place, properties and relationships common to all other classes.

More specifically, the class Artefact is showed in order to describe the **isDocumentedBy** relationship type, that links each model object to a Document.

* * *

The high-level UML diagram of the previous modelling rules follows.


Presentation
Artefact
Document
+ name
+ description
+ isNote
+ isLink
+ isDocument
* - isDocumentedby
*
DomainElement
+ Domain.name
+ value
+ description
- composedBy
- hasUserView
*
DataCube
+ name
+ description
+ type
- hasCompatibilityRule
0..1
CompatibilityRule
+ name
+ description
+ Variable.name[n]
+ includeExclude
+ value[n]
1..*
- parent-child
1
- hasStructure
*
Variable
+ name
+ description
+ Domain.name
Domain
+ name
+ description
+ algebraicStructure
- belongsTo
DataElement
+ name
+ variable
+ useDomain
+ isDimension
+ isMeasure
+ isAttribute
+ isTimeDimension
+ isEntityDimension
+ isUnit
+ isDecimal
+ isPrecision
1..*
- involve
- hasVariable
1
- definedOn
- refersTo
*
DomainSubset
+ name
+ description
+ isFullset
+ isCoded
+ isFiniteSet
- qualify
- hasDomainSubset
1
- subsetOf

3. THE XBRL "PROFILE"

The identified XBRL "profile" is compliant with the modelling rules depicted in paragraph 2, although not all of them have been expressed in XBRL terms. Possible future variations of those rules, and consequently of the "profile" itself, would be intended as incremental, thus ensuring backward compatibility.

The "profile" is deemed to be compliant with the criteria depicted in paragraph 1. In fact, according to Banca d'Italia experience in primary reporting, the "profile" is able to handle whatever reporting schema modelled applying the rules of paragraph 2.

A preliminary aspect of the "profile" is to choose the optimal way to represent data with a classification structure. As SPEC recognises, all data are "naturally" multi-dimensional, because they have at least a measure, a time and an entity to whom the measure refers. In fact, SPEC offers a standard way to deal with data having a time dimension and an entity dimension. Moreover, SPEC suggests a possible way, based on XBRL "segment" and "scenario" elements, to deal with data having dimensions other than those explicitly identified. This suggested way has been taken into account in this note, together with other possible ways to represent multi-dimensionality in full compliance with SPEC. All these different alternatives have been identified and commented, and one of them has been proposed. Having done that, an XBRL "profile" is described that incorporates the proposed solution for multi-dimensionality and, to the possible extent, the previously defined modelling rules.

It is worth noting that in the following one-dimensional data must be intended as data whose measure is identified by time and entity only, while multi-dimensional data must be intended as data whose measure is identified by time, entity and some other dimensions.

3.1 The multi-dimensional identification

Four different techniques have been identified and their compliance with the criteria depicted in paragraph 1 has been evaluated.

Item-based technique

This technique tends to transform a multi-dimensional structure in a one-dimensional one. It has been cited for sake of completeness and should not be really taken into account, because this choice would be disastrous, from the data administration perspective (i.e. "manageable taxonomy" criterion), in case of variables defined on domains with a high cardinality. At the same time, it is worth noting that this technique is essentially the only one, within those identified, that makes full use of the Calculation linkbase.

Segment/scenario-based technique

Each measure inside a DataCube would be represented as an XBRL item; the classification structure of the DataCube would be specified into the XBRL "period", "entity" and "segment" elements; the DataCube attributes not explicitly handled by SPEC would be specified into the XBRL "scenario" element.

The possible advantages of this technique would essentially be full compliance with FRTA 2.3.4 rule, where the use of "segment" and "scenario" elements is suggested, although not imposed (SHOULD), for multi-dimensional representation. This indication has recently been reinforced in FRIS 1.0 document, paragraph 2.6.

The envisaged disadvantages are explained by the same FRIS document, where there is written: <<The contents of the segment and scenario elements are practically unconstrained by XBRL: any elements not from the XBRL instance namespace are allowed …>>. This aspect has in fact some

implications: (i) there is no formal mechanism, based on an XBRL taxonomy, to communicate to the users the classification structure of a multi-dimensional data; (ii) there is no way to verify with a standard XML-aware software that the segment/scenario content referenced by the context of a multi-dimensional data is really compliant with the multi-dimensional structure of that data[7]. In short, this way to represent multi-dimensional structures is deemed not to comply to the "effective description" criterion and to the "software optimisation" criterion.

Tuple-based technique

Each DataElement, excluding those explicitly handled by SPEC, would be defined as an XBRL item. Each DataCube would be defined as an XBRL tuple, composed by all XBRL items previously defined. The XBRL instance document would then contain tuples only.

The envisaged advantages of this technique would be: (i) efficiency in data administration, because of the possible reuse of XBRL items in many DataCube definitions; (ii) formal definition of the multi-dimensionality; (iii) possible use of off-the-shelf XML tools for the structural validation of multi-dimensionality.

The possible disadvantages of this technique, besides the partial conflict with FRTA 2.3.4 rule, would be; (i) Calculation linkbase could never be used; (ii) all data would be represented as tuples and this could be a little bit misleading in cases, like e.g. financial reporting, where data is essentially one-dimensional.

Tuple/item-based technique

The same principles of the previous technique would apply, except that one-dimensional data would be represented as XBRL items.

The envisaged advantages would be the same of the tuple-based technique; for what refers to disadvantages, besides the partial conflict with FRTA 2.3.4 rule, Calculation linkbase could not be used for multi-dimensional data only.

The proposed solution

As we mentioned before, the item-based technique would be discarded. The segment/scenario-based technique would be discarded as well, because of its inability to describe and check the right association between a multi-dimensional data and its classification structure. The tuple-based and the tuple/item-based techniques seem to comply to all criteria in paragraph 1.

The tuple/item-based technique would be chosen.

3.2 The XBRL "profile"

In what follows, rules are given to translate in XBRL terms the data model depicted in paragraph 2, according to the chosen tuple/item-based technique.

3.2.1 DomainSubset

A FullSet (i.e. a **DomainSubset** having **isFullset** property equal "true") would be represented as an XSD ComplexType, derived by restriction from the appropriate XBRL built-in type.

The xsi:name assigned to the XSD ComplexType would be equal to the **name** property value of the FullSet. The **description** property would be put into the ComplexType\annotation\documentation element[8].

For uncoded FullSet and for coded FullSet with an infinite set of codes, the restriction may specify which facets can be applied to the base type (i.e. the XBRL built-in type).

[7] An ad-hoc software, that knows what structure is expected for each multi-dimensional data, should be used instead.
[8] More specifically it is the XSD "documentation" element, contained into the XSD "annotation" element appearing at the beginning of a ComplexType definition.

For coded FullSet with a finite set of codes, the restriction should specify the list of enumeration.

Other **DomainSubset** (i.e. DomainSubsets having isFullSet=false) would be translated into XSD ComplexType derived by restriction from the XSD ComplexType representing the FullSet. All the rules specified for FullSet would apply also to other DomainSubsets[9].

3.2.2 DomainElement

A **DomainElement** would be a single XSD enumeration value listed into the XSD ComplexType definition corresponding to the DomainSubset to whom the DomainElement belongs. The **value** property would be mapped into the xsi:value attribute of the XSD enumeration element. The **description** property would be placed inside the XSD enumeration\annotation\documentation element[10].

The **parent-child** relationship type between DomainElements would currently not be mapped into the proposed "profile", due to supposed SPEC inability to manage enumeration of ComplexType.

3.2.3 Variable

A **Variable** would be represented by an abstract XBRL item[11]. The xsi:name of the item would be equal to the **name** of the Variable. The xsi:type of the item would be equal to the FullSet name describing all the possible values that the Variable itself can assume.

The association between a Variable and the corresponding **description** property would be represented by means of a standard label into the Label linkbase.

It could be useful to distinguish in a taxonomy abstract items representing Variables from other abstract items used e.g. for presentation purposes. That could easily be addressed through the usage of Definition linkbase. As a matter of fact, all abstract items representing Variables could be grouped into a custom definition link having role "is-a-variable".

3.2.4 DataElement

For each **DataElement** (having isTimeDimension=false, isEntityDimension=false, isUnit=false, isDecimal=false, isPrecision=false) an XBRL item would be defined. This item would have the attribute abstract="false" and would be in the substitution group of the abstract item representing the Variable identified by the **name** property.

The xsi:type attribute would be set to the DomainSubset name represented by the **useDomain** property value.

The time dimension (DataElement where **isTimeDimension**=true) would be captured inside the period element, contained into the context element.

The entity dimension (DataElement having **isEntityDimension**=true) would be captured inside the entity element, contained into the context element.

The unit attribute (DataElement having **isUnit** =true) would be captured inside the unit element.

The decimals and precision attributes (DataElement having **isDecimal**=true or **isPrecision**=true) would be represented respectively by means of decimals and precision attributes of the item.

The **qualify** relationship type between different DataElements would be captured into a specific definition link with a custom arcRole.

---

[9] The **subsetOf** relationship type would not be explicitly represented in the given profile. In fact, this relationship is implicitly described through the ComplexType derivation mechanism.

[10] More specifically it is the XSD "documentation" element, contained into the XSD "annotation" element, which is itself contained in the XSD "enumeration" element.

[11] Its purpose would be to define a reusable XBRL item to be referred from inside a DataCube definition.

3.2.5 DataCube

In order to better clarify the criteria used to translate a DataCube in XBRL terms, a distinction would be made between a SimpleDataCube and a ComplexDataCube.

A SimpleDataCube is a DataCube with the following characteristics:

- has only one measure;
- has a time dimension;
- has an entity dimension;
- if numeric, has a unit attribute and (possibly) decimals and precision attributes;
- has no additional Variables.

A SimpleDataCube is defined by means of the XBRL item that represents its measure.

A ComplexDataCube is a DataCube that has more than one measure and/or additional Variables. It would be represented by means of an XBRL tuple. The name of the tuple would be the **name** property of the DataCube. The association between the DataCube and the corresponding **description** property would be represented by means of a standard label into the Label linkbase.

The **hasStructure** relationship type would be captured by the tuple content; more specifically:

- for each DataElement playing the role of dimension (different from time dimension and entity dimension) a child element of the tuple would be declared. This child element would be mandatory (i.e. it would have xsi:minOccurs attribute=1 and xsi:maxOccurs attribute=1) and it would reference (xsi:ref attribute) the XBRL item corresponding to that DataElement;
- for each DataElement playing the role of measure, a child element of the tuple would be declared. This child element would reference (xsi:ref attribute) the XBRL item corresponding to that DataElement;
- for each DataElement playing the role of attribute (different from unit, decimals and precision attributes), a child element of the tuple would be declared. This child element would reference (xsi:ref attribute) the XBRL item corresponding to that DataElement;
- time dimension and entity dimension are captured by the XBRL context element;
- unit attribute is captured by the XBRL unit element;
- decimals and precision attributes are captured by the XBRL decimals and precision attributes.

The role played by each DataElement into a DataCube (i.e. **isDimension, isMeasure, isAttribute**) would be captured by means of a definition link having a custom role value of "DataCube-structure". The definition link would define an arc from the XBRL item corresponding to the DataElement up to the tuple corresponding to the DataCube. The arcRole value would be a custom one and it would represent the role played by the DataElement into the DataCube. The following arcRoles would be defined: "isDimension", "isMeasure", "isAttribute".

The **composedBy** relationship type that refers to ComplexDataCubes has no correspondence in SPEC, while it is expressed by Calculation linkbase in case of SimpleDataCubes.

The **hasCompatibilityRule** relationship type has not been translated in XBRL terms. There are some options to translate it, but a confrontation with SPEC experts is deemed necessary.

3.2.6 CompatibilityRule

The **CompatibilityRule** class has not been translated in XBRL terms. There are some options to translate it, but a confrontation with SPEC experts is deemed necessary.

3.2.7 Presentation

The **Presentation** class and the **hasUserView** relationship type would be mapped into the XBRL items, tuples and Presentation linkbase, according to the XBRL standard way of functioning.

3.2.8 Artefact

The **Artefact** class has not been translated in XBRL terms.

Nevertheless, the i**sDocumentedBy** relationship type may be represented by means of reference links, according to the XBRL standard way of functioning.

3.2.9 Document

The **Document** class maps into the XBRL reference element contained inside an XBRL reference link element.

4. CONCLUSIONS

The proposed "profile" is deemed to meet all the criteria depicted in paragraph 1 and its semantics is completely specified.

With reference to its use within the "translation" project, the group would start from the templates and, for each template, it would:

- identify the domains (i.e. Exposure class, Exposure type, CRM type and so forth), the domain elements and the variables;
- define the data in terms of dimensions, measures and attributes, together with the domain subsets that apply to each of them within a DataCube.

In XBRL terms, the taxonomy would contain the definition of:

- all the ComplexType corresponding to the identified domain subsets;
- the abstract items corresponding to the variables;
- the items corresponding to the dimensions, measures and attributes;
- the tuples corresponding to the multi-dimensional DataCubes;
- the abstract items necessary for presentation purposes;
- all the necessary linkbases.

From the perspective of a European supervisor willing to adopt the taxonomy, but with a customisation requirement, the following could be said:

- for what refers to B category (see CEBS document "New Solvency Ratio – Toward a Common Reporting Framework", November 2004, point 14) the supervisor would only select the list of DataCubes of interest;
- for what refers to C category, the supervisor would extend the COREP taxonomy through the following additional definitions:
  - new ComplexTypes, that refer to the domain subsets coherent with the chosen level of detail;
  - new items that refer to the COREP variables and to the new ComplexTypes;
  - new tuples that make use of these items;

- a “similar-tuple” definition link, linking each new tuple to the corresponding COREP tuple.